

Editorial

AI

Künstliche Intelligenz (englisch: Artificial Intelligence, AI) ist derzeit **das** Zukunftsthema der Informatik. Dabei ist es uralte: Geprägt wurde der Begriff schon 1956. Damals versuchte man mit formaler Logik z. B. mathematische Sätze zu beweisen oder Knobelaufgaben zu lösen. Andere Ansätze verwendeten (z. B. medizinische) Expertensysteme und durchsuchten deren Datenbank nach Empfehlungen, oder gingen *brute force* vor, indem sie alle Reaktionsmöglichkeiten durchprobierten und bewerteten – wie der berühmte IBM-Schachcomputer „[Deep Blue](#)“, der am 10.02.1996 den amtierenden Schachweltmeister Gary Kasparov schlug. Doch ließen sich diese Ansätze nicht verallgemeinern, da sie auf bestimmte Probleme zugeschnitten waren.

Der Durchbruch gelang schließlich „Neuronalen Netzen“, die ganz ähnlich wie das menschliche Gehirn funktionieren: Angelernt mit Beispieldaten korrigieren sie die „Gewichte“ der Eingänge in ihren miteinander verknüpften „Knoten“ so lange, bis das Ergebnis den Erwartungen entspricht. Die Lernqualität hängt damit natürlich von den Lerndaten ab: Je mehr und je aussagekräftiger, desto verlässlicher löst das neuronale Netz eine Aufgabe. Dieser auch „maschinelles Lernen“ (ML) genannte KI-Ansatz wurde erst mit heutigen Rechnern effizient möglich – und erobert derzeit immer neue Anwendungen, denn er lässt sich für praktisch alles nutzen, was ein Computer „üben“ kann. Der Nachteil: Wie beim menschlichen Gehirn weiß man nicht, was das neuronale Netz genau gelernt hat – und es kann zu Fehlentscheidungen kommen.

Dirk Fox, Stefan Falk

Eine solche KI-Anwendung, die vor zwei Jahren durch einen [Artikel des Guardian](#) Furore machte, ist GPT-3 – ein Text-Erzeugungs-Algorithmus der Firma [OpenAI](#). Gefüttert wird er mit Inhalten aus dem Internet – und erzeugt zu einem ausgewählten Thema kurze Notizen bis hin zu ganzen Büchern, die sogar an unterschiedliche Schreibstile angepasst werden können.

Die Wahrheit eines Sachverhalts kann GPT-3 nicht prüfen und ist daher auf eine Bewertung der Plausibilität angewiesen. Was aber könnte plausibler sein als eine sehr oft wiederholte Behauptung? Wenn (was sicher bereits passiert) immer mehr Texte von einem solchen Automaten erzeugt werden, wird die Verbreitung (und damit wiederum die Plausibilität) häufiger Behauptungen verstärkt – und Widerlegungen, die es schon heute schwer haben (wie z. B. der Mythos vom [hohen Eisengehalt von Spinat](#)), dürften keine reelle Chance haben. Ähnliches könnte für neues Wissen gelten – denn das hat ja gerade die Eigenschaft, selten publiziert worden zu sein.

Immerhin könnt ihr sicher sein, dass (auch) diese ft:pedia von den Autoren „handverfasst“ ist – denn sie hält exklusives Wissen für euch bereit. Und damit unsere Kinder die Grenzen von KI-Systemen verstehen, sollten sie damit experimentieren – **und** die ft:pedia lesen...

Beste Grüße,
Euer ft:pedia-Team

P.S.: Am einfachsten erreicht ihr uns unter ftpedia@ftcommunity.de oder über die Rubrik *ft:pedia* im [Forum](#) der ft-Community.

Inhalt

AI	2
Spielerei mit Rundbausteinen	5
Maxizahnräder – Modul 1,5 oder lieber 16?	7
Großprojekt Seilbahn (Teil 5): Bewegung.....	10
Club-Modelle Teil 1: 1979/3	19
Die Praktika von fischertechnik im Jahr 2022.....	36
Die Kunst der H-Brücke: ... and roll!	45
Silberlinge: Original oder Nachbau (Teil 8).....	53
Elektronik-Module (Teil 8): Silberlinge = Elektronikmodule?!...	62
Silberlinge: Flip-Flop in Hardware oder Software?	72
Sensoren am TXT: Farberkennung mit der Kamera.....	81
Einführung in ftScratch (3): Der Barcodeleser	85
Android App für den BT-Empfänger mit dem MIT App Inventor 2	93
TX-Light: Arduino (Uno/Mega) und ftDuino aus ROBO Pro ansteuern.....	104
TX-Simulator für den ftDuino.....	110

Termine

Was?	Wann?	Wo?
Süd-Convention	01.-02.10.2022	Fördertechnik-Museum Sinsheim
Clubdag NL	29.10.2022	Schoonhoven

Gedenken

Am 03.04.2022 ist Dirk Haizmann nach langer Krankheit gestorben. Er war über 30 Jahre im Deutschlandvertrieb das „Gesicht“ von fischertechnik und hat die ft Community, das Forum und die Conventions nach Kräften unterstützt. Wir werden ihn nicht vergessen.

Impressum

<http://www.ftpedia.de>

Herausgeber: Dirk Fox, Ettlinger Straße 12-14,
76137 Karlsruhe und Stefan Falk, Siemensstraße 20,
76275 Ettlingen

Autoren: Arnoud van Delden, Stefan Falk, Dirk Fox, Hans-
Christian Funke, Holger Howey, Thomas Kaiser, Peter Krijnen,
Thomas Magin, Kurt Mexner, Rüdiger Riedel, Tilo Rust.

Copyright: Jede unentgeltliche Verbreitung der unveränderten
und vollständigen Ausgabe sowie einzelner Beiträge (mit
vollständiger Quellenangabe: Autor, Ausgabe, Seitenangabe
ft:pedia) ist nicht nur zulässig, sondern ausdrücklich erwünscht.
Die Verwertungsrechte aller in ft:pedia veröffentlichten Beiträge
liegen bei den jeweiligen Autoren.

Tipps & Tricks

Spielerei mit Rundbausteinen

Rüdiger Riedel

Die Rundsteine 15×15 und 30×15 sind eine enorme Bereicherung zur Gestaltung der fischertechnik-Modelle. Als Anregung habe ich hier ein paar „Mandalas“ zusammengestellt.





Getriebe

Maxizahnräder – Modul 1,5 oder lieber 16?

Rüdiger Riedel

fischertechnik-Zahnräder haben Modul 1,5 und in den Getrieben 0,5. Braucht jemand für Demonstrationszwecke richtig große Zahnräder? Mit Modul 16? Hier sind sie.

Zwei zusammengefügte Bausteine 15×30 rund ([163202](#), [163439](#), [172544](#) oder [173043](#)) erinnern an die Zähne unserer fischertechnik-Zahnräder. Da muss nur noch ein passendes Grundgerüst her.

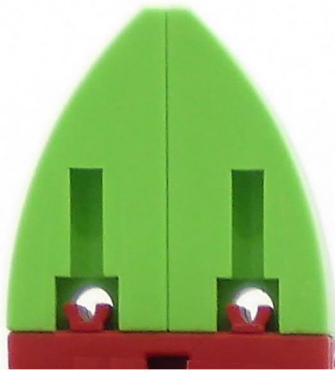


Abb. 1: Der Maxizahn

Die Ähnlichkeit mit den Zahnrädern nach [1] ist nur oberflächlich. Deshalb müssen wir den Abstand der Zahnräder zueinander ausprobieren, sodass sie einigermaßen sauber ineinander kämmen.



Abb. 2: Zweimal Z10

Wir beginnen mit dem Z10

Der Ring wird aus zehn Baugruppen BS7,5 + WS15 + BS5 + WS7,5 + BS5 + WS15 plus 10 Federnocken ([31982](#)) in den BS7,5 zusammengesetzt. Die Winkelsumme ist dann 375° statt der erforderlichen 360°, aber der Ring lässt sich problemlos schließen. In jeden der BS7,5 kommt außen eine Federnocke, in jeden zweiten ebenfalls auf der Innenseite. Mit strammem Sitz passt jetzt innen ein Speichenrad 90×15 ([19317](#) oder [36916](#)) hinein.



Abb. 3: Baugruppe

Je zwei Bausteine 15×30 rund werden mit zwei Federnocken an einen Baustein 5 15×30 3N ([38428](#)) gefügt und 10 von diesen Baugruppen mit dem Ring zum Z10 zusammengesetzt.

Nochmal das Ganze und wir können ein „Getriebe“ bauen. Unter die je drei BS30 und BS15 Loch kommt je ein BS5 oder Baustein 5 15×30 ([35049](#)). Damit können wir den Abstand zwischen den beiden Zahnrädern optimieren.

Kleines Zahnrad Z6

Auf eine Drehscheibe 60×5,5 ([31019](#)) schieben wir sechs BS7,5 und mit Federnocken befestigen wir sechs der „Zähne“ wie oben beschrieben.

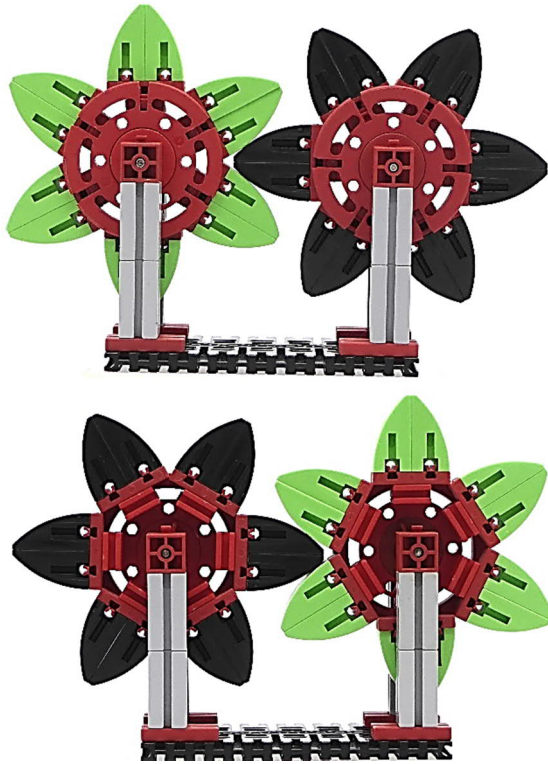


Abb. 4: Z6, Vorder- und Rückseite

Großes Zahnrad Z16



Abb. 5: Z16

Die Statikräder werden mit je 16 Riegelsteinen 15 ([32850](#)) bestückt und daran seitlich die Zähne aus je zwei Bausteinen 30 rund und zwei Federnocken befestigt.

Modulberechnung

Der Modul m berechnet sich aus dem Teilkreisdurchmesser (Wälzkreisdurchmesser) d geteilt durch die Anzahl der Zähne z [1].

$$d = z \cdot m$$

Somit

$$m = \frac{d}{z}$$

6 Zähne

Achsabstand $d = 96$ mm:

$$m = 96 \text{ mm} / 6 = 16 \text{ mm}$$

Also $m = 16$.

10 Zähne

Achsabstand $d = 163$ mm:

$$m = 163 \text{ mm} / 10 = 16,3 \text{ mm}$$

Also ungefähr $m = 16$.

16 Zähne

Achsabstand $d = 262$ mm:

$$m = 262 \text{ mm} / 16 = 16,375 \text{ mm}$$

Also ungefähr $m = 16$.

Zahnradkombinationen

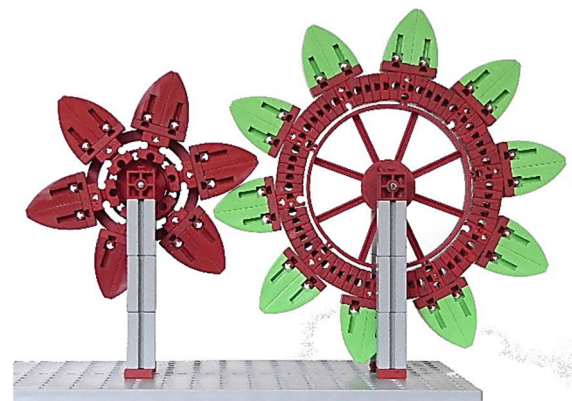


Abb. 6: Z6 auf Z10



Abb. 7: Z10 auf Z16



Abb. 8: Z6 auf Z16

Zahnstange

Schieben wir auf eine Reihe von BS30 Zähne gemäß Abb.1 im Abstand von 22,5 mm (ein BS15 plus ein BS7,5 als Abstandsmaß) und setzen wir unter die Stützen des großen Zahnrades statt der 5 mm hohen Bausteine je ein BS7,5, dann kämmt die Zahnstange recht gut.

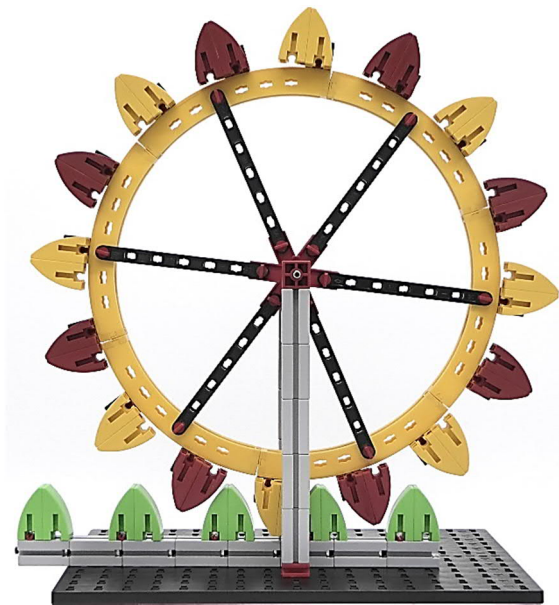


Abb. 9: Z16 mit Zahnstange

Nochmal zur Beachtung: Gegenüber der Abhandlung von Thomas Püttmann ist dieser Beitrag vor allem auf die Schönheit und Größe der Modelle gerichtet. Die Ermittlung der Module erfolgt durch Einstellung des Abstandes der Zahnräder nach Gefühl. Irgendeine Genauigkeit ist da nicht gegeben. Technische Anwendungen sind auch nicht das Ziel.

Quellen

- [1] Thomas Püttmann: *Zahnräder und Übersetzungen (Teil 3)*. ft:pedia 1/2012, S. 13–21.

Modell

Großprojekt Seilbahn (Teil 5): Bewegung

Tilo Rust

Diese Serie begleitet das Großprojekt „Kuppelbare Einseilumlaufbahn (10-MGD)“ im Förder-technik-Museum Sinsheim von Anfang bis zur Fertigstellung und Ausstellung auf der BUGA 2023 in Mannheim. In diesem Teil zeigen wir den aktuellen Stand des Modells und was sich jetzt schon bewegt.

Transportbewegung

Dass sich eine Seilbahn bewegen muss, ist klar. Dass aber eine Seilbahn bewegt wird, das geht nur im Modell. Denn das Modell soll an unterschiedlichen Orten aufgestellt werden. Zu diesem Zweck sind die Stationen, welche 5,00 m lang sind, in je zwei Module zerlegbar. Daraus ergeben sich letztlich Packmaße von 2,50 m × 1,40 m mit einer Höhe von 1,80 m. Diese Maße wurden gewählt, um noch durch eine Doppelflügeltür zu kommen und die Module quer auf einen LKW verladen zu können. Außerdem wurde der Unterbau für den Transport mit Hubfahrzeugen (Gabelstapler, Ameise etc.) vorgesehen und verfügt über Zurrmöglichkeiten für die Ladungssicherung. Über die Transportverschalung und -sicherung, an der wir gerade arbeiten, werden wir hier noch gesondert berichten. Das Gewicht der Module ist aber jetzt schon enorm und wird durch die weiteren Aufbauten, den Motor und das Spanngewicht noch erheblich anwachsen.

Jede Bewegung der Module soll aber im Betrieb ausgeschlossen sein. Daher verfügen die Module über Niederschraubfüße, welche auf Anti-Rutschmatten aufsetzen und zudem das Modell nivellieren. Schließlich werden beide Module einer Station mit von Doppelmayr/Garaventa extra angefertigten Zentrierzapfen (Abb. 1) verbunden und zueinander gezogen.

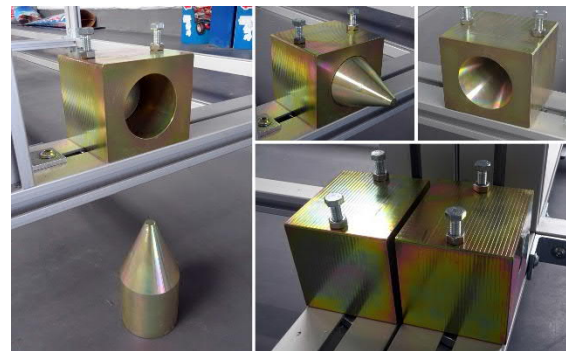


Abb. 1: Zentrierzapfen

Schwere und hochpräzise Zentrierzapfen von Doppelmayr/Garaventa werden verwendet, um die Module form-, kraftschlüssig und genau aneinander zu binden, so dass sie auch unter Erschütterungen keine Bewegung zulassen. Sie sind so gestaltet, dass der Zapfen zum Transport entnommen werden kann, da keine Teile über die Verschalung herausstehen dürfen.

Die zweite Ebene der Verbindung beider Module geschieht mit den Zentrierstücken aus Abb. 2, die in die Axmann-Industrieprofile greifen. Verspannt werden beide Module mit Gewinde-Zugstangen. So wird der verschiebbare Teil des jeweils hinteren Moduls ebenfalls bewegungsunfähig gemacht. Über die Planung dieser Teile und der Niederschraubfüße haben wir bereits in [5, ..., 8] berichtet.

Was aber für den Transport notwendig ist, nämlich der ausreichende Abstand aller Bauteile zur Transportverschalung, macht

nun ein Problem: Der Abstand zwischen den Stationen muss wieder geschlossen werden. Hierzu ist das jeweils hintere Modul mit einem Schlitten ausgestattet, der nach vorne geschoben werden kann. Dabei greifen von uns entwickelte Zentrierstücke in das vordere Modul und verbinden so den tragenden Bau (Abb. 2).

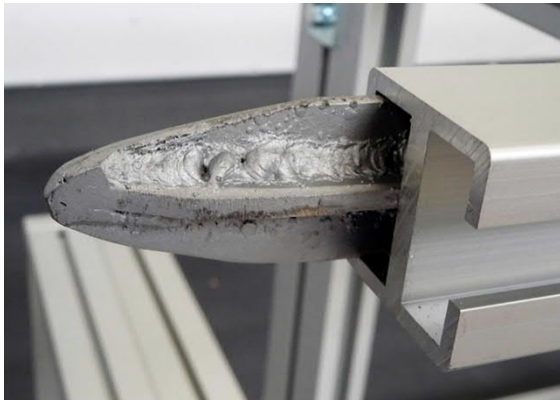


Abb. 2: Zentrierstücke

Anschließend werden die Laufschiene der Gondel, Niederhalteschiene und Reifen-

förderer mit wenigen Handgriffen mechanisch stabil und präzise zusammengefügt. Hier kommt der alte fischertechnik-Trick zum Zuge, mit Stangen zwei Bausteine ohne Zapfenverbindung genau aneinander zu fügen.

Wie genau die elektrische Verbindung hergestellt wird, das müssen wir noch klären. Ziel ist es jedenfalls, dass die Platzierung und Montage der Stationen sowie die Verbindung der Module in kürzester Zeit (wir sprechen von 30 Minuten pro Station) erfolgen kann, da Aufstellen, Auspacken, Ausrichten, Inbetriebnahme etc. noch genug Zeit in Anspruch nehmen und wir z. B. auf einer Ausstellung nicht viel Zeit dafür haben. Genau das aber ist der Grund, warum wir uns hier so viele Gedanken machen.

Umlaufbewegung

Wie im Original werden einlaufende Gondeln aus dem Seil ausgekuppelt und fahren dann auf einer Schiene langsam um die Station herum, bis sie auf der anderen Seite

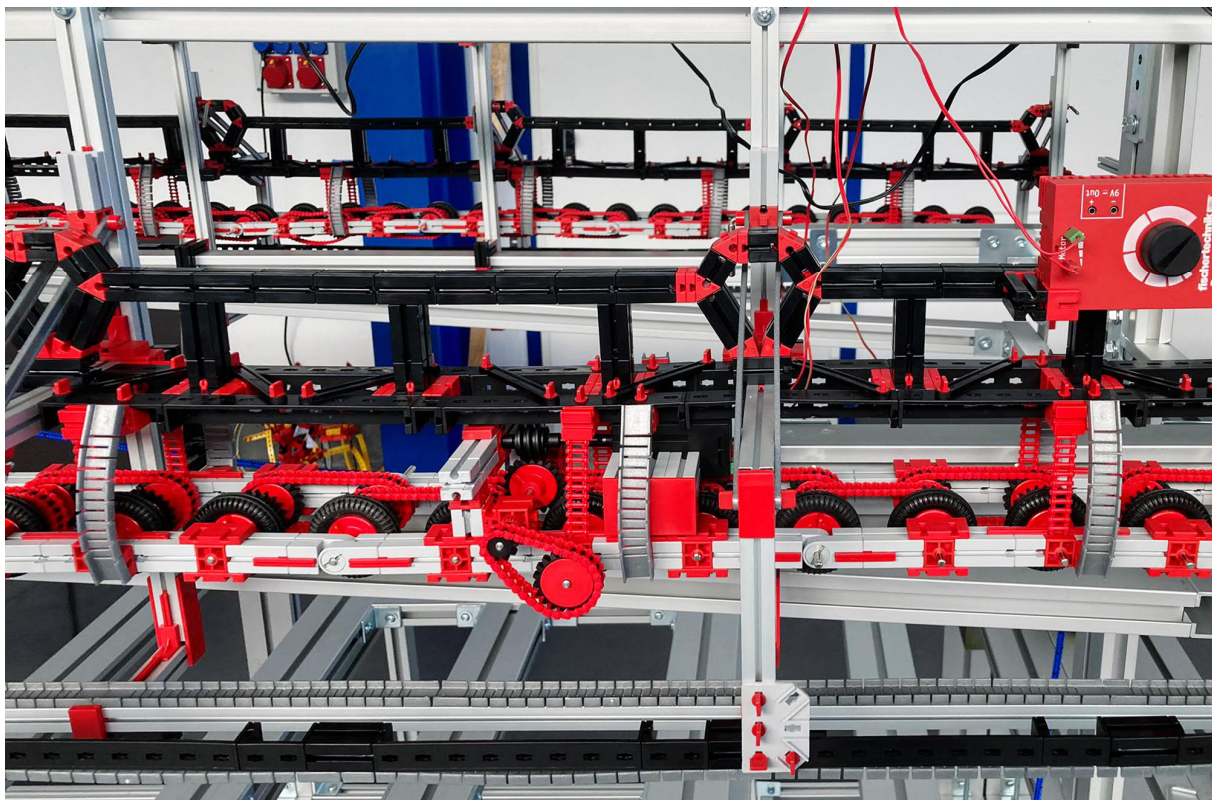


Abb. 3: Reifenförderer

wieder in das Seil eingekuppelt werden und auf die Strecke fahren.

Den Kuppelmechanismus und die Klemme haben wir bereits in vorangegangenen Beiträgen vorgestellt. Jetzt müssen wir von den Prototypen auf das fertige Modell umrüsten. Über eine insgesamt über 5 m lange Strecke transportieren über 150 Reifen die Gondel langsam an einem Laufschuh oberhalb der Klemme. Das muss absolut zuverlässig geschehen und darf sich nicht auseinanderschieben. Schließlich soll das Modell ein halbes Jahr auf der BUGA23 täglich viele Stunden mehr oder minder nonstop fahren. Alle Bauteile müssen nicht nur genau platziert werden, sondern auch miteinander verschränkt und verriegelt sein.

Zum Reifenförderer (Abb. 3): Nun wird es ziemlich komplex. Die Gondeln, die später auf der Flexschiene (unterer Bildrand)

laufen, werden von oben hängend mit den Reifen angetrieben und so durch den Stationsumlauf befördert. Diese Reifenförderer gliedern sich in Module und müssen in der Geschwindigkeit aufeinander abgestimmt sein. Der Antrieb ist dabei möglichst zuverlässig zu bauen, sodass er die Dauerbelastung aushält. Gleichzeitig sind die C-Stücke (etwas rechts von der Bildmitte mit der Knotenplatte am unteren Ende) so zu gestalten, dass sie die Last stabil tragen und sich nicht dynamisch verziehen. Dank vieler Versuche läuft nun alles.

Im Original-Reifenförderer (Abb. 4) sind die C-Stücke und die Trägerkonstruktionen, die auch wir im Modell nachgebildet haben, gut zu sehen. Alle Reifen sind hier mit Keilriemen verbunden; wir setzen das mit unserer fischertechnik-Kette um.



Abb. 4: Reifenförderer im Original

Im Gegensatz zum Original haben wir zudem den Nachteil, dass unsere Reifen nicht wirklich nachgeben. Im Original ist die Eindrücktiefe 3 cm – das wären bei uns 3 mm. Um das zu erreichen, wäre ein großer Anpressdruck nötig. Da wir diesen nicht aufbringen können und wollen, ist unsere Toleranz viel kleiner. Oder anders formuliert: Über 5 m müssen 150 Reifen bis auf einen Millimeter genau justiert werden und dürfen sich dann nicht mehr bewegen. Und das, obwohl die Laufschiene darunter sich leicht verbiegt (auch wenn es ein fischertechnik-Alu-Profil ist) und sich der ganze Aufbau durch wechselnde Lasten immer wieder etwas verzieht.

Nach stundenlangem Tüfteln direkt am Modell ist es nun gelungen, alles so weit zu stabilisieren und zu verstreben, dass die erste Gondel sauber durch die Station läuft. Dabei wird ausgekuppelt, abgebremst und um die Kurve gefahren, beschleunigt und wieder eingekuppelt.

Der Stationsumlauf besteht aus Modulen des Reifenförderers wie in Abb. 5 gezeigt. Der Antrieb ist so platziert, dass sein Gewicht für den Anpressdruck sorgt, aber er mittig im Modul ausgerichtet ist. Dieses kann sich vertikal durch die alten Gelenkbausteine 45 bewegen und wird durch die gebogenen Flexschiene 90 gehalten. Die lose Kette neben der Flexschiene begrenzt den Durchhang. Längs kann es (fast) frei

schwingen, um Stöße abzufangen. Das hier gezeigte Modul hat eine Übersetzung von 1:40 und treibt acht Reifen an – für den Umlauf sind über 150 nötig...

Knifflig ist hierbei die Übersetzung. Bei Einfahrt in die Station läuft das Seil (und damit auch die Gondel) mit einer Geschwindigkeit von 60 – 65 cm/s. Nach dem Auskuppeln wird diese dann in zwölf Stufen auf nur 3 – 4 cm/s abgebremst. Hierzu müssen die Reifenförderer über selbstgedruckte Z24-Ritzel verfügen, die das Verzögern stufenweise vornehmen. Doch diese müssen mit XM-Motoren angetrieben werden. Nach Ermittlung der Drehzahlen haben wir eine passende Übersetzung (1:1 direkt ab Welle) für das erste (schnelle) Modul gefunden. Für die langsameren Module und den Weitertransport in der Station benötigen wir mehrere Getriebe, die schließlich auf 1:40 untersetzen. Damit können wir auch gut die Reibungsverluste in den Griff bekommen, die über die lange Strecke auftreten. Wir tüfteln noch, aber so wie es aussieht, müssen mindestens 10 XM-Motoren in der Geschwindigkeit aufeinander abgestimmt werden. Über die elektronische Seite werden wir ebenfalls noch ausgiebig berichten.

Aktueller Stand Ende August 2022: Die Reifenförderer wurden mit Antrieb einem eintägigen Dauertest unterzogen – keinerlei Fehler oder Ausfälle.

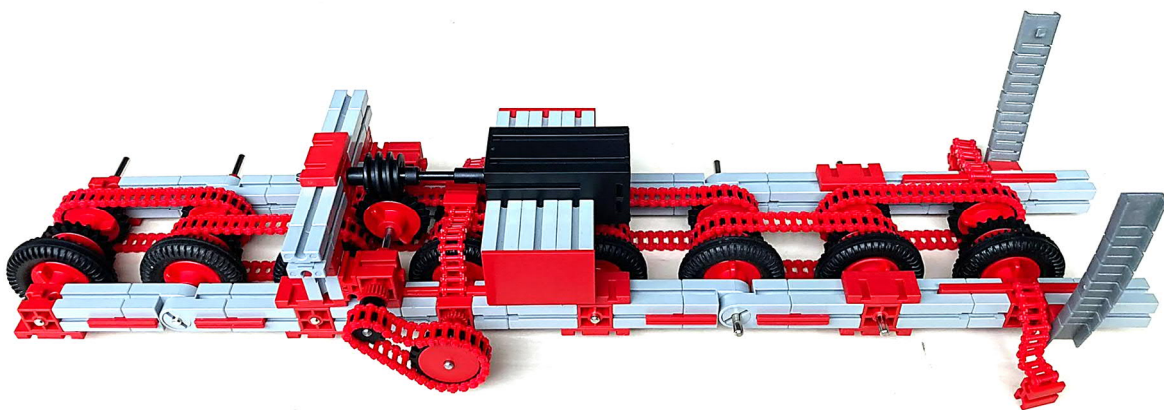


Abb. 5: Ein Modul des Reifenförderers

Spannbewegung

Eine Bewegung in der Seilbahn ist fast unsichtbar und auch in den Originalen sehr versteckt:

Um das Seil ständig gespannt zu halten, gleichzeitig aber Lastschwankungen auszugleichen, ist mindestens eine der großen Seilscheiben auf einem Wagen montiert, der in der Station entlang der Längsachse verschoben werden kann und damit das Seil strafft. In unserem Modell besitzen wir (wie auch im Original der Doppelmayr-/Garaventa-Bahn für die BUGA23) zwei Spannwagen, je einen pro Station. Wir können damit einen Spannweg von 70 cm (im Original 7 m) zurücklegen. Wie im Original ist die Umkehrstation, Bergstation „Luisenpark“, für die Grundspannung des Seils zuständig und die Antriebsstation, Talstation „Spinelli“, für die dynamischen Lasten.

Das bedeutet, dass das Seil zuerst mit Hilfe der Luisenparkstation auf Spannung gebracht und diese Einstellung im laufenden Betrieb nicht mehr verändert wird. Das bewerkstelligen wir mit Hilfe eines Seilzuges, der mit einer Winde und Verriegelung den Spannwagen nach hinten zieht.

Wenn nun die Gondeln auf das Seil gehängt werden (bzw. später wie im Original automatisch aus der Garage fahren), dann erhöht sich der Zug am Seil. Es hängt stärker durch, und der Spannwagen in Spinelli wird nach vorne gezogen. Hier bekommt das Spanngewicht Bedeutung. Es muss so austariert werden, dass einerseits der Durchhang nicht zu groß wird (und der Wagen an den vorderen Anschlag fährt), andererseits die Zugkraft nicht zu groß wird, so dass die Anlage zerstört würde. Ein Riss des Seils ist ausgeschlossen; es kann über 21 kN aushalten, während wir gerade mal mit 0,5 kN spannen. Dazu müssen wir das Spanngewicht in gewissen Stufen erhöhen. Hierzu legen wir Gehwegplatten als Gewichte auf einen Aufzug, der sich in einem Schacht unter der Anlage versteckt. Da wir aber nur

30 cm Hubweg zum Ausgleich der dynamischen Bewegungen zur Verfügung haben, können wir mit Hilfe eines zweiten Seilzuges nachjustieren, der genau verriegelt und in beide Richtungen gebremst werden kann. Schließlich kommt es nur darauf an, dass die richtige Kraft im Fahrbetrieb den Spannwagen und den Aufzug jeweils in der Mitte der Schienen hält.

Gruppenbewegung

Wer kräftig arbeitet, darf auch kräftig feiern. Deswegen hat unser Team eine Einladung, zu unserem Hauptsponsor Doppelmayr/Garaventa nach Wolfurt in das Hauptwerk zu kommen, gerne angenommen. Eine große Delegation durfte einleitenden Worten von Herrn Schütte aus dem Marketing folgen und bekam so die aufregende Firmengeschichte und neuesten Projekte des Seilbahnherstellers aus Österreich zu sehen. Anschließend hatten wir Zugang zu den Produktionshallen. Die Augen wurden immer größer, als wir Schweißroboter und Fertigungsstraßen sehen konnten, die unsere kühnsten Vorstellungen übertrafen. Es vergingen nur wenige Minuten im Materiallager, bis der Spruch „alles unter 120 mm Dicke nennen wir hier ‚Blech‘“ zum Kult wurde.



Abb. 6: Eine Doppelmayr-Totpunktclasp

Eine originale Doppelmayr-Totpunktclasp (Abb. 6) so nah zu sehen und alle technischen Fragen beantwortet zu bekommen, ist für Modellbauer ein Hochgefühl.

Die Klemme wiegt zwar mehr als ein ganzes Modul unserer Anlage, kann aber auch Gondeln mit einer Tonne Gewicht plus zehn Fahrgäste halten. Täglich, bei Wind und Wetter, Sturm und Vibration, zuverlässig, absolut sicher. Das müssen wir mit fischertechnik erstmal hinbekommen.

Herr Eck, der Museumsleiter des Förder-technik-Museums Sinsheim, konnte nun seinen Part an unserem gemeinsamen Projekt vorstellen und erläutern, dass Fördertechnik mit fischertechnik gut zusammenpasst und deswegen die Workshops so gut besucht sind. Dass eine Seilbahn, zumal vom Weltmarktführer, in die Ausstellung gut hineinpassen würde, ließe sich mit unserem Modell zumindest ein wenig realisieren.

Leider konnten Vertreter der fischertechnik-Werke nicht an unserer Reise teilnehmen; ein Input auch von diesem Sponsor wäre schön gewesen. Das lässt sich aber sicher beim nächsten Besuch nachholen.

Schließlich waren wir an der Reihe und konnten nicht nur unser Projekt im Detail vorstellen, sondern auch einen Ausblick auf unsere kommende Arbeit geben. „Wenn unser Team bis heute diese Leistung in nur 600 Stunden erbracht hat, was können wir dann erreichen, wenn wir zwei- oder drei-

tausend erbracht haben?“ Das stimmt nachdenklich, hatten die Zuhörer doch einen viel höheren Arbeitsaufwand geschätzt, als der bis jetzt erbrachte (Stand Mai 2022).

Die mitgebrachten fischertechnik-Modelle (Gondeln, Reifenförderer, Klemme und Rollenbatterie, sowie Mini-Modell) wurden fachlich diskutiert und Ideen und Tipps ausgetauscht.

Ein Highlight war die Vorführung einer Animation, welche unsere Unterkonstruktion und die Montage der Module zeigt, sowie die Premiere unseres eigenen Musikstücks, das wir nun bei jedem Video verwenden können, komponiert von unserem Projektmitglied Alexander Salameh. „Up and Around“ genießt man am besten mit der Animation „Talstation“ auf unserem YouTube-Kanal [2].

Abschluss fand die Reise bei einer Einladung zu einem gemeinsamen Abendessen, für welche wir uns hier noch einmal herzlich bedanken möchten. Dieser Besuch stärkt unser Team bis heute und zeigt die Wertschätzung unserer Arbeit.

Anfangsbewegung

Unser Modell in der Öffentlichkeit zu zeigen ist natürlich ein wichtiger Punkt. Auf dem Maimarkt in Mannheim konnten wir das schon einmal tun. Das Förder-technik-

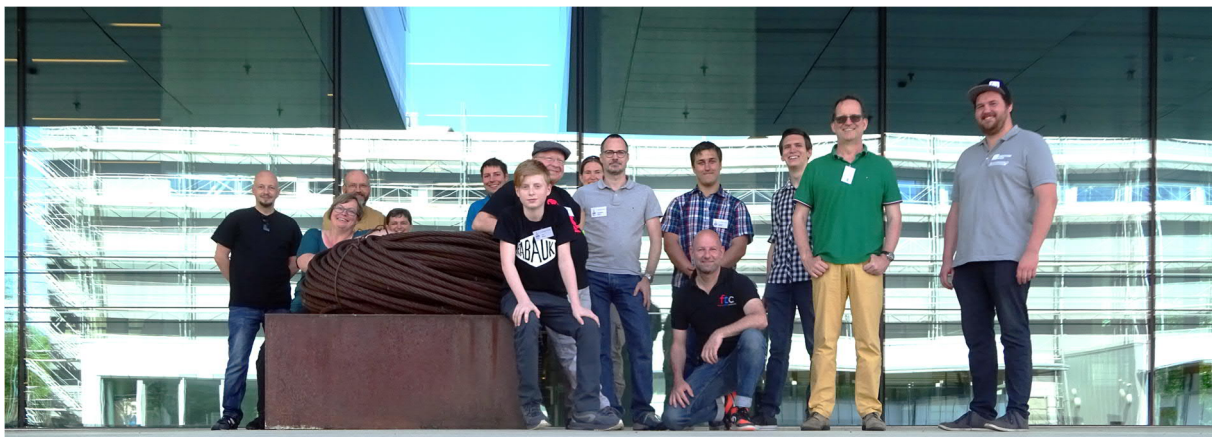


Abb. 7: Die 14-köpfige Delegation vor dem Firmensitz der Doppelmayr/Garaventa-Gruppe in Wolfurt. Die Skulptur aus einem alten Seil zeigt gut, mit welchen Dimensionen hier gearbeitet wird. Eine spannende Zeit im wahrsten Sinne des Wortes.

Museum hatte hier einen Stand in Halle 3. Doch auch das Rhein-Neckar-Fernsehen (RNF) hatte Sendezeit für uns und der Stand der BUGA23 gab Gelegenheit, das Original- und 1:10-Modell nebeneinander zu sehen.

Die Abb. 8 bis 11 geben einen Eindruck davon. Leider haben wir keine Aufnahme der Sendung des RNF. Aber sicher wird das nicht das letzte Interview gewesen sein.



Abb. 8



Abb. 9



Abb. 10



Abb. 11

Großbewegung

Auch in Mannheim bewegt sich etwas bei der großen Seilbahn der BUGA23 [1]. Dank der Projektleiterin Frau Haas durften wir die Baustelle besuchen und den Aufbau der Anlage aus nächster Nähe sehen. Das Einheben des Spannwanne dort dauerte nicht lange – war aber umso spannender und faszinierend. Stand September 2022 stehen bereits neun der zehn Stützen, und die Station „Spinelli“ ist fast fertig. Ein Video dazu haben wir auf unserem YouTube-Kanal [2] veröffentlicht.

Aufregend, beim Bau des Originals dabei sein zu dürfen! In Abb. 12 wird in der Spinelli-Station der Spannwanne in die Station gehoben. Gut zu sehen ist, wie die Schienen zwischen dem Steher 1 (links) und 2 (rechter Bildrand) schräg abfallen. Das haben wir im Modell ebenfalls getreu umgesetzt; es ist nötig, damit das Seil unter der Klemme abtauchen kann. Die Seilscheibe wird an die rote Nabe geflanscht und der Antrieb sitzt oben auf. Siehe hierzu auch unser Video „BUGA vs. Modell“ auf unserem YouTube-Kanal [2].



Abb. 12: Die Baustelle auf der BUGA23

Der nächste Besuch wird uns dann auf unser eigenes Ausstellungsgelände im Luisenpark, unweit der dortigen Seilbahnstation führen. Den Baufortschritt und den Transport unseres Modells im März 2023 werden wir natürlich auch dokumentieren.

Aber auch auf der ftc-Süd-Convention 2022 werden wir ausstellen. Zwar müssen wir die Anlage dafür nur wenige Meter bewegen,

aber bis dahin wollen wir noch einiges gebaut haben.

Weiterbewegen

Wir sind sehr in Zeitverzug und arbeiten jedes Wochenende im Museum [3] am Modell. Helfende Hände, vor allem jetzt im Herbst, werden benötigt – bitte melden [4]!

Die Aufgaben? Die zweite Station (Luisenpark, mit über 5000 Teilen) in Angriff nehmen, dazu den Unterbau aus Industrieprofilen zusammenschrauben, die nächsten 25 Gondeln (je 250 Teile) bauen, weitere drei Rollenbatterien (mit je 900 Teilen) erstellen oder den Steuerstand (mit ft-Pi und ftDuino) zum Laufen bringen, die Transportverkleidung schreinern, die Anlage verkabeln, die Seilführung optimieren, alles testen und debuggen – und schließlich zur BUGA23 bringen und sechs Monate täglich betreuen (hier suchen wir dringend noch Helferinnen und Helfer für alle Termine).



Abb. 13: Ein Teil des Teams vor dem Modell der Talstation – vielen Dank für eure bisherige und zukünftige Arbeit. Weitere Helferlein sind gesucht – bitte melden!

Kenntnisse in fischertechnik sind sinnvoll, aber nicht unbedingt nötig, Das Alter spielt keine Rolle, Nähe zu Sinsheim oder Mannheim hilft. Bautage sind immer samstags von 11-17 Uhr im Fördertechnik-Museum [3].

Außerdem suchen wir noch Spender, die 3D-Druckteile für uns erstellen. Dateien liefern wir, da wir für alle 50 Gondeln Spezialteile benötigen.

Danke an die (neuen) Helferinnen und Helfer – gemeinsam bewegen wir etwas.

Quellen

- [1] BUGA23 - Bundesgartenschau in Mannheim: <https://www.buga23.de/>
- [2] [YouTube-Kanal](#) des Projekts
- [3] [Fördertechnik Museum Sinsheim](#), Untere Au 4, 74889 Sinsheim (direkt neben dem Technik-Museum)
- [4] Kontakt zur Projektgruppe „Seilbahn“, Projektleiter Tilo Rust, Schifferstadt: ft.seilbahn@gmail.com
- [5] Tilo Rust: *Großprojekt Seilbahn (Teil 1): Von der Idee zum Kick-Off*. [ft:pedia 1/2021](#), S. 56–65.
- [6] Tilo Rust: *Großprojekt Seilbahn (Teil 2): Erste Elemente*. [ft:pedia 2/2021](#), S. 45–53.
- [7] Tilo Rust: *Großprojekt Seilbahn (Teil 3): Die Stationen*. [ft:pedia 3/2021](#), S. 28–37.
- [8] Tilo Rust: *Großprojekt Seilbahn (Teil 4): Fundamentale Arbeiten*. [ft:pedia 1/2022](#), S. 37–43.

Elektronik

Club-Modelle Teil 1: 1979/3

Hans-Christian Funke

Im letzten Heft hatte ich angekündigt, Modelle aus den alten Club-Heften von fischertechnik nachzubauen und im neuen Gewand zu präsentieren. Die Auswahl der Modelle stelle ich chronologisch beginnend ab 1979 rückwärts – also in die Vergangenheit – vor.

Einleitung

Die Serie beginne ich mit dem Club-Modell aus dem Heft 1979/3 – ein Spielautomat. Das ist schon der zweite von fischertechnik vorgestellte Spielautomat. Im Heft 1975/2 wurde der erste Spielautomat vorgestellt – vollkommen anders aufgebaut und mit einer anderen Schaltung. Die zweite Version ist – finde ich – allerdings schöner, sowohl von der Optik als auch von der technischen Umsetzung (Mechanik und Elektronik), obwohl ähnliche Elemente und Baugruppen enthalten sind (Geldausgabe, Drehscheiben).

Ein Blick auf das Titelbild der Bauanleitung zeigte mir schon gleich, dass mich einige Herausforderungen erwarten würden. Einige Teile wie z. B. das Schneckengetriebe gibt es nicht mehr, auch nichts Vergleichbares, um mit einer neueren Lösung zu arbeiten, wie z. B. bei Kegelhauptzahnradern. Für die Elektronikbausteine (Silberlinge) können die Elektronikmodule eingesetzt werden, die sämtliche erforderlichen Funktionen zur Verfügung stellen.

Vorbereitungsarbeiten

Zunächst musste ich mir selbst erst einmal die aktuellen Teile bei Franz Santjohanser [1] bestellen, der dankenswerter Weise die Projektreihe mit unterstützt und einen großen Teil der Bauteile gesponsert hat – an

dieser Stelle noch einmal vielen Dank dafür.

Die meisten Teile aus dem Modell sind identisch mit den aktuellen Teilen, häufig nur in anderen Farben. Daher können die Modelle in der Regel 1:1 nachgebaut werden. Es ist auch nicht mein Ziel die Modelle neu zu konstruieren, sondern nur mit neuen Teilen nachzubauen. Obwohl es mich beim Nachbauen an einigen Stellen in den Fingern gejuckt hat, die Umsetzung anders zu gestalten, überlasse ich das gerne jedem selbst, wenn er das Modell nachbaut.

Natürlich muss an verschiedenen Stellen nach einer anderen Lösung für die Realisierung gesucht werden – hier ist dann der Designer und Konstrukteur gefragt. Bei der Schaltung konnte ich dann aber nicht mehr an mich halten und habe eine kleine Änderung eingeführt, weil die abgedruckte Schaltung in Zusammenspiel mit der Konstruktion von Schalter mit Schaltscheiben einen kleinen Fehler aufweist, der zwar selten auftritt, aber den Spieler um seinen Gewinn brächte, wenn dieser Fall eintritt (Erläuterung des Fehlers im nächsten Abschnitt).

Damit jeder das Modell nachbauen kann, muss man die Originalbauanleitung zur Hand nehmen. Diese findet ihr beim fischertechnikclub NL [2]. Dort sind der Aufbau und die Funktionsweise des Modells beschrieben.

Umsetzung mit Hindernissen

In der Originalbauanleitung gibt es ein paar Stellen, an denen der Baumeister auf sich gestellt ist, weil die Fotodokumentation nicht genau genug alle Details wiedergibt, um die Konstruktion nachzubauen. Aber auch die mangelnde Qualität der abgelegten Bauanleitung ließ ab und zu die Identifizierung der Konstruktion oder der Bauteile nicht zu. An diesen Stellen musste ich improvisieren und ausprobieren, wie die Konstruktion gedacht war oder welche sich an dieser Stelle als zweckmäßig erweist. Die zusätzlichen und die nicht nachvollziehbaren Bauabschnitte habe ich hier in diesem

Artikel dokumentiert, sodass man zum Aufbau des Modelles beide Anleitungen heranziehen sollte.

Bevor ich mit dem Aufbau beginnen konnte, musste ich mir im Vorfeld ein paar Gedanken zu den alten bzw. neuen Teilen machen. Wie ersetze ich die alten Teile und durch welche neuen? Ist dafür eine andere Konstruktion erforderlich, und welche Änderungen zieht dies für die Gesamtkonstruktion nach sich?

Es gibt vier gravierende Änderungen in diesem Modell, die die Gesamtkonstruktion beeinflusst haben:

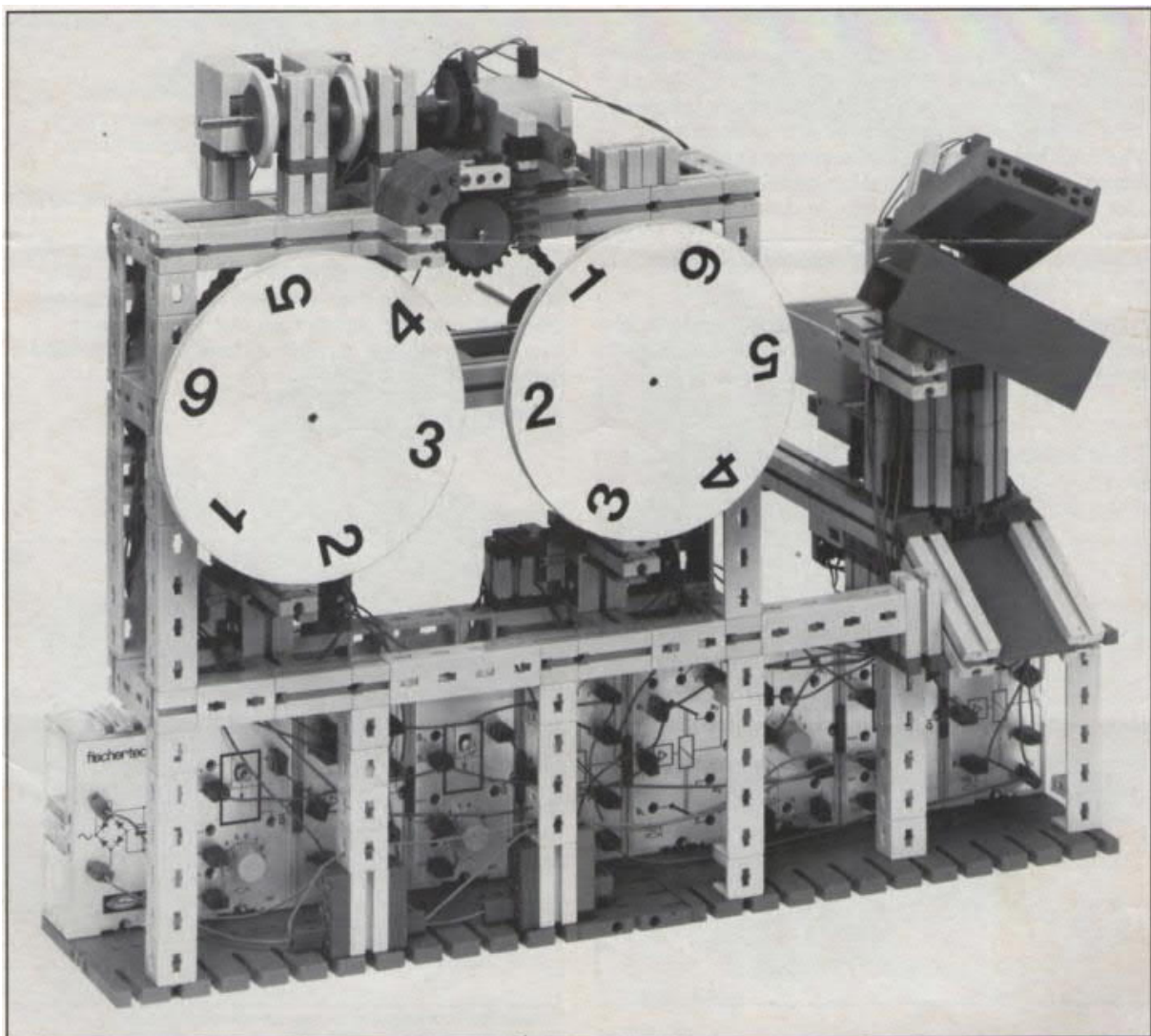


Abb. 1: Spielautomat (Version 2) aus der Bauanleitung 1979/3

1. Die Grundplatten,
2. die Elektronikmodule,
3. die alten Schleifringe und
4. der Antrieb (es gibt keine Schneckengetriebe mehr).

Die alten roten Grundplatten gibt es schon lange nicht mehr und als Alternative stehen

- die große Grundplatte 390×270,
- die Grundplatte 285×186,
- die kleine Grundplatte 120×60 und
- die kleine Grundplatte 120×90

zur Auswahl. Im Prinzip kann jede eingesetzt werden – die kleinen Grundplatten sollten vielleicht miteinander verbunden werden. Für dieses Modell habe ich mich für die Grundplatte 120×90 entschieden, die ich jeweils mit zwei Bauplatten 15×30 verbunden habe.

Die Elektronikbausteine (Silberlinge) wurden mit ihrer Bauform in die Konstruktion integriert und somit musste ich das Modell auf eigene Füße stellen. Die vier notwendigen Elektronikmodule befinden sich ebenfalls unter dem Modell, aber das Modell hat jetzt auf der Rückseite zusätzliche Winkelträger bekommen, auf denen die Konstruktion ruht.

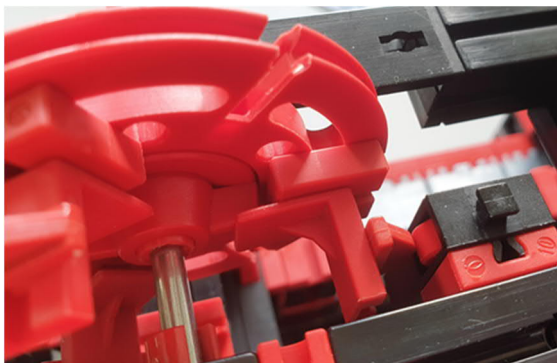


Abb. 2: Drehscheiben mit nachgestellten Noppen zum Betätigen des Tasters

Die Schleifringe aus den em-Baukästen sind ebenfalls schon lange nicht mehr in der Teilewelt zu finden, obwohl diese ab und an noch in Modellen auftauchen. Wie nun einen Ersatz schaffen? Ein Griff in die Trickkiste hat eine einfache und simple,

aber gleichwertige Lösung zu Tage gefördert (siehe Abb. 2).

Beim Antrieb für die Drehscheiben kam der gleichwertige XM-Motor (M1) zum Einsatz, der statt des Schneckengetriebes nur eine Achse zur Verfügung stellt. Auf diese Achse habe ich ein Zahnrad Z20 mit einer Flachnabe montiert. Mit leicht veränderter Position des XM-Motors konnte das Z20 direkt in die Zähne des Z20 vom Modell greifen und der Antrieb war hergestellt (Abb. 3).

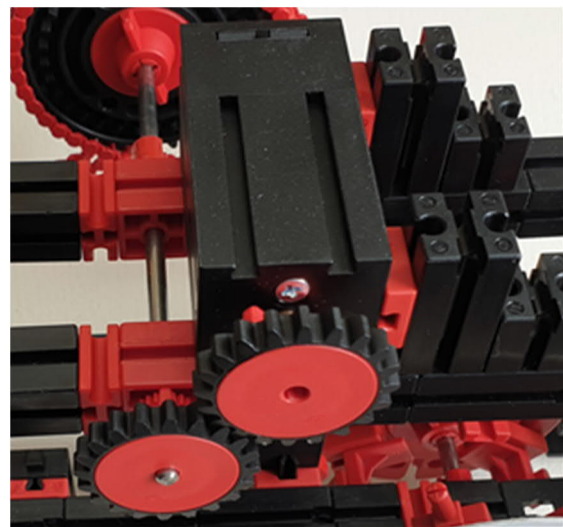


Abb. 3: Antrieb des Spielautomaten mit einem XM-Motor

Im Originalmodell werden über ein weiteres Schneckengetriebe zwei Schaltscheiben mit Schaltern betrieben. Hierfür habe ich eine Lösung mit Kegelzahnradern vorgesehen und aufgebaut.

Die Elektronik und Verdrahtung ist der vorletzte Teil beim Aufbau des Modells – letzter Teil ist die Abstimmung der Elektronik und Mechanik aufeinander. So hatte ich mich erst später mit der Schaltung beschäftigt. Nach ihrer Betrachtung und Analyse stellte ich fest, dass die Taster an den Schaltscheiben nur eine Ersatzkonstruktion darstellten. Der Designer der Schaltung hätte eigentlich noch ein Mono-Flop benötigt, um die Funktion des Spielautomaten optimal nachzubilden. Sowohl aus Platzgründen, denn die Reihe der Silberlinge war

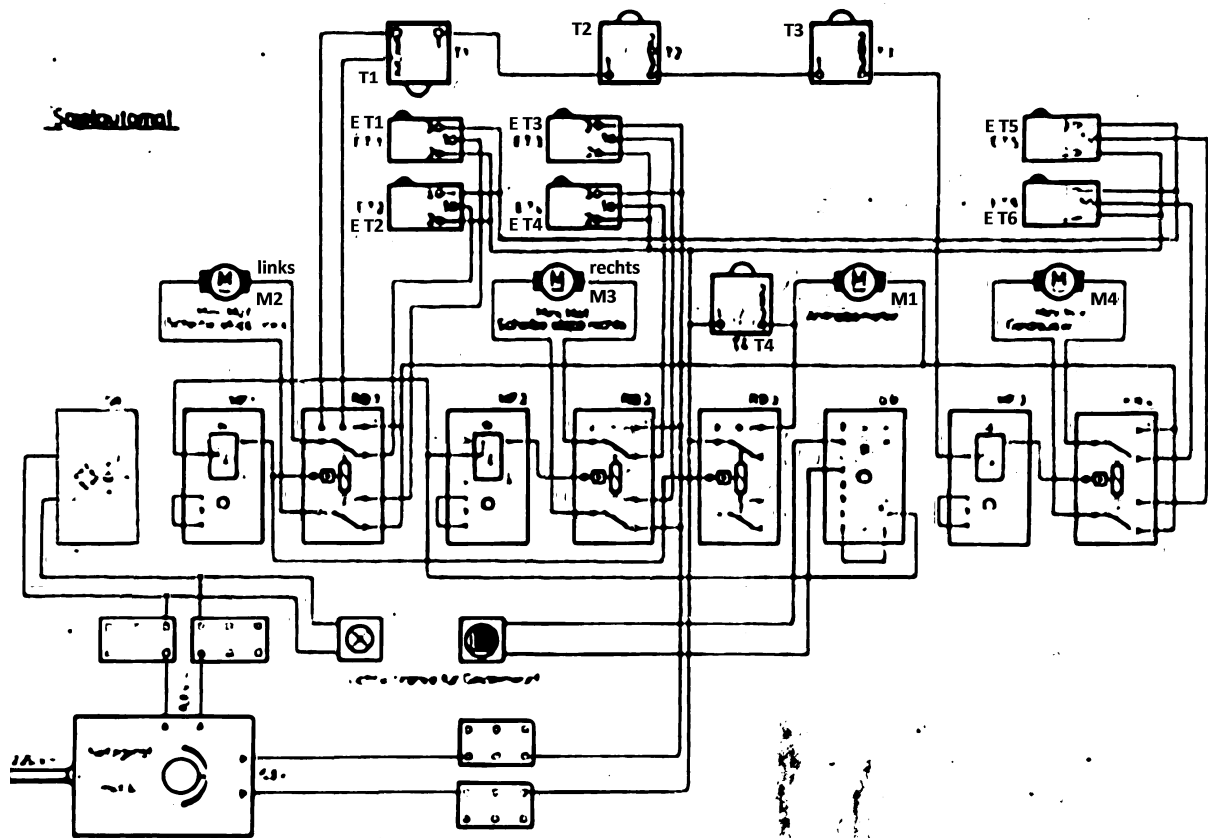


Abb. 4: Der Ausdruck des Schaltplans aus der Originalbauanleitung (leider sehr schlechte Qualität), etwas aufgewertet durch Retuschierung

schon voll, und andererseits aus Kostengründen wurde dieses Mono-Flop (MF) durch eine Schaltscheibenkonstruktion ersetzt.

Die Schaltscheiben an Taster T4 überbrücken das Relais für den Motor M1, der die Drehscheiben antreibt. Wird das Relais für den Motor M1 abgeschaltet, läuft der Motor durch die Überbrückung von Taster T4 weiter und soll für genügend Zeit sorgen, damit die linke Drehscheibe durch den Motor M3 gestoppt werden kann. Darüber hinaus soll über den Taster T1, der durch die zweite Schaltscheibe betätigt wird, ein Signal an das Mono-Flop für die Geldausgabe gegeben werden, wenn die Gewinnbedingung erfüllt ist (zwei gerade Zahlen nebeneinander = Taster T2 und T3 durchgeschaltet).

Aber genau hier liegt der anfangs erwähnte kleine Fehler: Wird das Relais für den

Motor M1 abgeschaltet und die Schaltscheibe, die Taster T4 bedient, befindet sich gerade an der Stelle, an der eben keine Überbrückung des Relais erfolgt (Lücke bei den Schaltscheiben), drehen sich auch die Schaltscheiben nicht mehr und der Taster T1 kann keinen Gewinn signalisieren, selbst wenn die Gewinnbedingung erfüllt ist.

Besser wäre ein weiteres Zeitintervall eines MF, das die Laufzeit der Drehscheiben steuert (Relais von Motor M1). Das Zeitintervall müsste dann enden, wenn der Motor M2 die linke Drehscheibe sicher gestoppt hat und damit das Ergebnis der Drehscheiben feststeht.

Ein großer Vorteil der Elektronikmodule ist, dass die meisten Funktionen doppelt vorhanden sind. So stehen auf einem Mono-Flop-Modul 70005 zwei unabhängige MFs zu Verfügung. Es werden laut Schaltplan drei MFs benötigt; somit bleibt ein MF frei.

Auf Basis der vorangegangenen Überlegung habe ich die Schaltung dahingehend verändert, dass das vierte MF das benötigte Zeitintervall für die Ansteuerung des Relais für den Motor M1 erzeugt. Durch diese Änderung wird der Taster T4 nicht mehr benötigt. Der Taster T1 soll mit Hilfe der Schaltscheiben einen Impuls an das MF für die Geldausgabe senden, sofern die beiden Taster T2 und T3 durchgeschaltet sind.

Und damit sind wir beim nächsten Problem, denn die Elektronikmodule bestehen aus digitalen Komponenten und reagieren auf kleinste Impulse (im Bereich von Nanosekunden). Wenn zum Beispiel die Drehscheibe rechts bereits steht und eine gerade Zahl zeigt (Taster T3 geschlossen), dann kann es beim weiteren Rotieren der Schaltscheibe und der linken Drehscheibe passie-

ren, dass auch noch Taster T1 und T2 kurzzeitig zugleich geschlossen werden und damit ein Impuls an das MF für die Geldausgabe gesendet wird, was eine unberechtigte Geldausgabe zur Folge hätte. Aus diesem Grund sollte der Impuls für die Geldausgabe von der Schaltung generiert werden, wenn die Drehscheiben zum Stillstand gekommen sind. Dies bedeutet wiederum, dass der Taster T1 ebenfalls nicht benötigt wird und damit die gesamte Konstruktion mit dem Antrieb der Schaltscheiben hinfällig ist.

In Abbildung 5 ist die neue Schaltung mit den beschriebenen Änderungen zu sehen. Es fällt sofort auf, dass die Anzahl der Elektronikmodule, die für die Realisierung der Schaltung erforderlich sind, deutlich geringer ist (nur halb so viele wie Silberlinge).

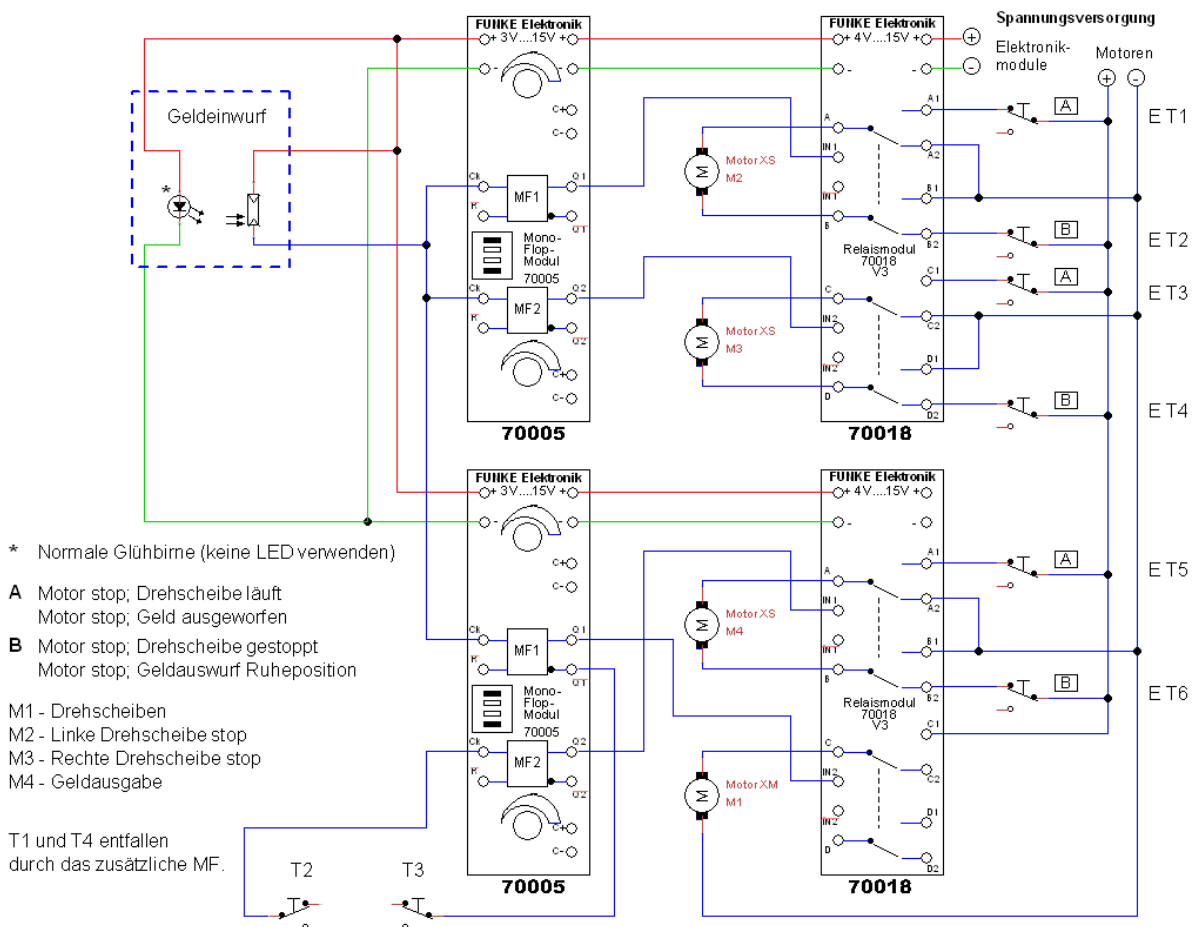


Abb. 5: Der angepasste Schaltplan mit Elektronikmodulen für den Spielautomaten

Aufbau des Spielautomaten

Jetzt kommen wir zum eigentlichen Aufbau des Spielautomaten. Hier kann im Wesentlichen der originalen Bauanleitung gefolgt werden und ich gebe hier weitere Hinweise und Tipps für den Aufbau.

Bei diesem Modell werden drei Sonderteile benötigt, die nicht zum fischertechnik-Sortiment gehören. Diese werden für eine so genannte Rutschkupplung benötigt sowie für die Drehscheiben, die aus Pappe oder Sperrholz angefertigt werden können. Die Teile erhält man im nächsten Baumarkt:

1. Zwei Dichtungsringe $\frac{1}{2}$ " (Zoll). Es kann auch ein Gummiring mit mindestens 4 mm Innendurchmesser verwendet werden. Er sollte aber flach sein wie in Abb. 6 (rechts) zu sehen. Mein Dichtungsring hatte einen größeren Innendurchmesser; dort habe ich eine passende Unterlegscheibe (U-Scheibe) eingesetzt mit einem Innendurchmesser von 4 mm und einem Außendurchmesser von 15 mm (Abb. 7).
2. Vier Unterlegscheiben \varnothing 4 mm (innen). Diese gibt es im fischertechnik-Einzelteilesortiment ([161261](#)), aber die beiden U-Scheiben für die Gummiringe sollten gleich passend im Baumarkt besorgt werden. Die U-Scheiben brauchen einen Innendurchmesser von 4 mm, wie in Abb. 6 (links) zu sehen.



Abb. 6: U-Scheibe und Dichtungsgummi $\frac{1}{2}$ "



Abb. 7: Dichtungsgummi mit U-Scheibe

3. Zwei Scheiben mit \varnothing 100 mm aus Pappe oder Sperrholz. Für den Spielautomaten

habe ich einfach die Pappe der Rückseite eines Schreibblocks verwendet, die Pappe mit einem Blatt Papier beklebt und auf der Rückseite – also auf die Pappe – mit Hilfe eines Zirkels (Topf oder Glas geht auch) einen Kreis von 100 mm gezeichnet und ausgeschnitten (Abb. 8). Bei Sperrholz verwendet man am besten 4 mm-Sperrholz aus Pappel, das man ebenfalls im Baumarkt bekommt. Auf das Papier kommen zum Schluss noch die Zahlen, entweder die Selbstklebenden aus dem fischertechnik-Einzelteilesortiment ($2 \times$ [143389](#)) oder aus der Vorlage bei den Downloads zu diesem Beitrag [3].

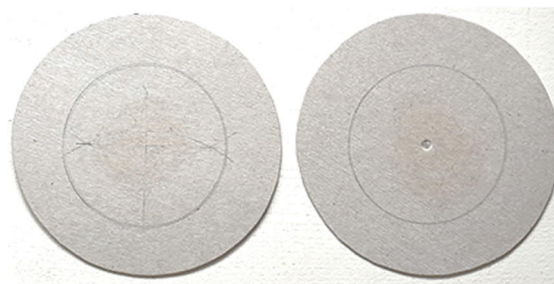


Abb. 8: Zwei Scheiben mit Durchmesser 100 mm aus Pappe von einem Schreibblock

Bei den Scheiben empfehle ich, mit einer Metallachse in die Mitte der Pappscheiben eine Mulde zu drücken (siehe rechte Scheibe in Abb. 8). Bei Verwendung von Sperrholz muss mit einem Bohrer 4,0 mm genau in der Mitte gebohrt werden (nicht durchbohren).

Beim Aufbau des Spielautomaten lässt man die 170-mm-Achse etwa 1 mm aus der Flachnabe herausgucken, sodass die Bohrung oder die Mulde genau auf der überstehenden Achse ihren Platz findet. So sind die Scheiben genau in der Mitte platziert und eiern nicht beim Drehen.

Tipp: Die genaue Mitte des Kreises liegt dort, wo das Einstichloch der Zirkelspitze ist. Ist das nicht mehr zu sehen, kann man zwei Messungen an einer Stelle der Scheiben mit der größten Abmessung durchführen – idealerweise im Winkel von 90° Grad zueinander – und die jeweilige Mitte markieren. Der Schnittpunkt der beiden Markierungen ist der gesuchte Mittelpunkt der Scheibe. Wer möchte und es noch aus der Schulzeit weiß, kann die Mitte auch konstruktiv mit Zirkel und Lineal ermitteln.

Die Unterkonstruktion

In Abb. 9 ist die Konstruktion mit den Grundplatten 120×90 mm und den Halterungen für die Elektronikmodulen zu sehen.

Das Modell kann mit zwei Schaltnetzteilen ([505287](#) mit Anschluss über ein Versorgungsmodul 70003) betrieben werden, mit einem Schaltnetzteil mit Power-Regler

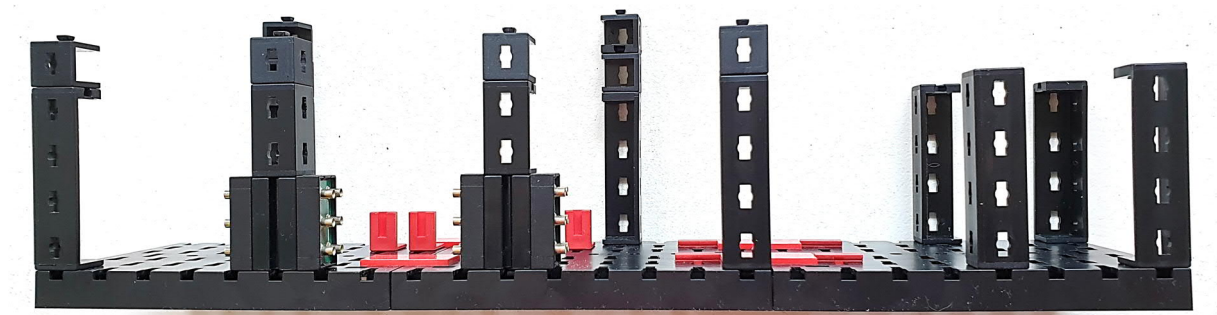


Abb. 9: Unterkonstruktion mit Grundplatten 120×90 mm

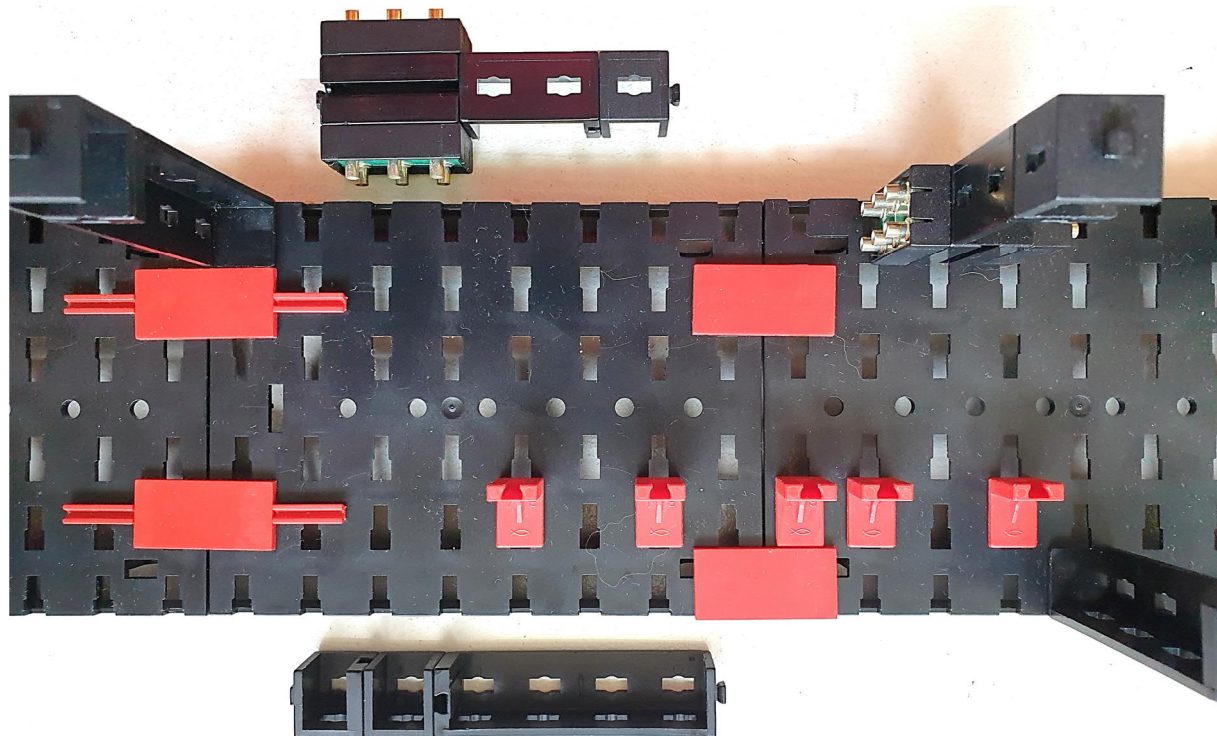


Abb. 10: Unterkonstruktion mit Halterung für die Elektronikmodule

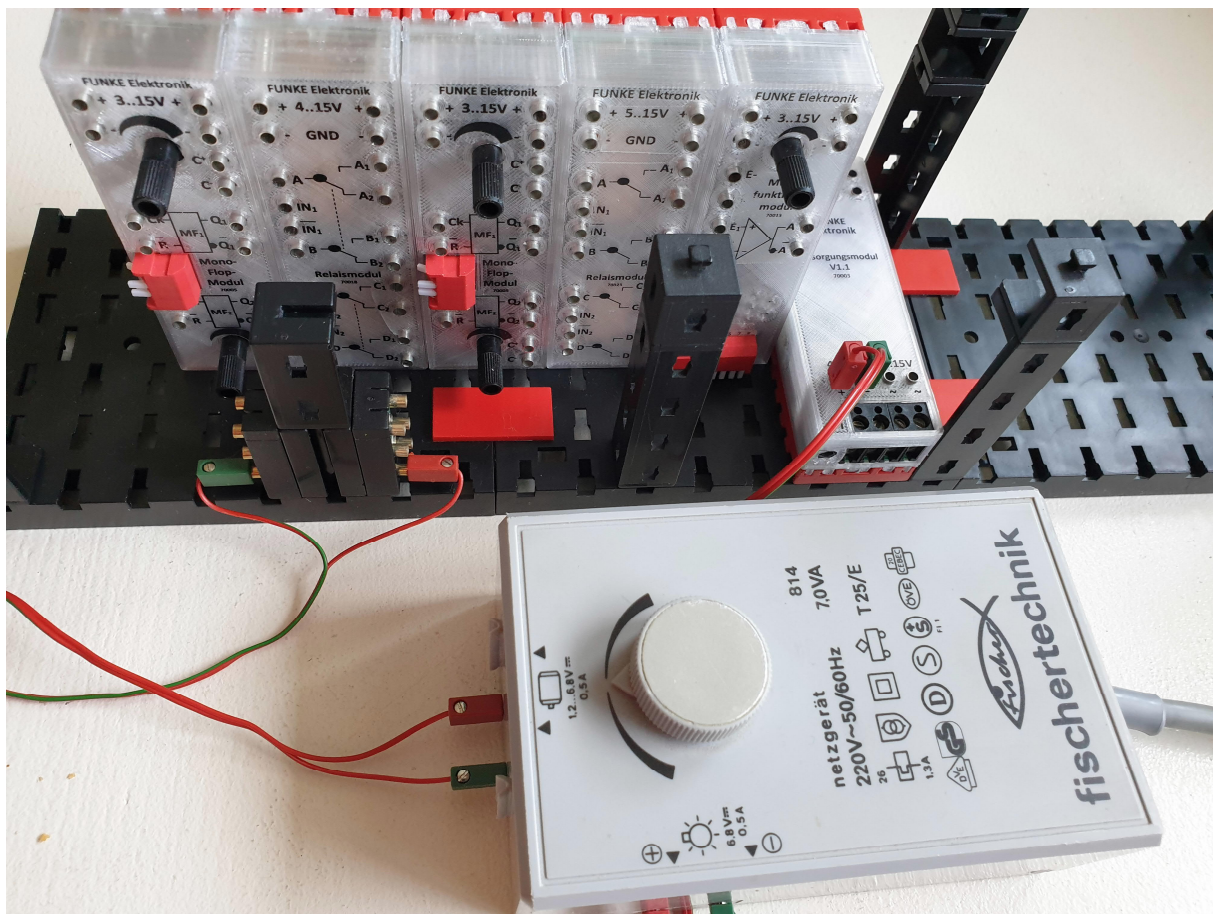


Abb. 11: Versorgung mit einem alten Trafo mot4 über ein Versorgungsmodul 70003

([139778](#)), mit einem alten Trafo mot.4 zusammen mit einem Elektronikmodul 70003 oder mit zwei Akku-Packs. Die Motorenversorgung sollte von der Versorgung der Elektronik getrennt werden.

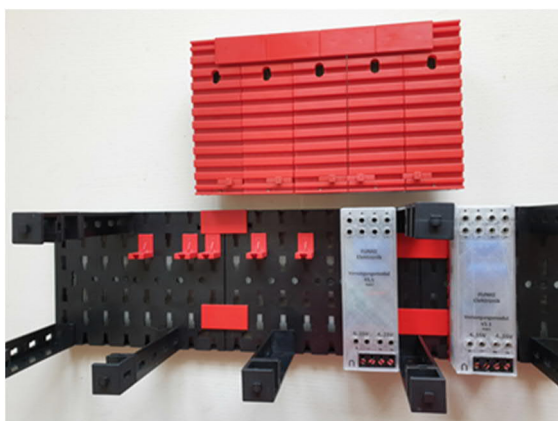


Abb. 12: Es können bequem zwei Versorgungsmodul untergebracht werden, falls zwei Schaltnetzteile ohne Power-Regler verwendet werden.

Die Verteilerplatten eignen sich gut, um einen Übergabepunkt für die Versorgungsspannungen zu haben, besonders bei Anschluss über einen Trafo oder Power-Regler. Akku-Packs können gut rechts und links der Elektronikmodule platziert werden.

Aufsatz für die Stopper-Motoren

Bei dieser Baugruppe habe ich nur zwei Federnocken als Stopper für die Zahnstangen hinzugefügt. Meine Zahnstangen waren so leichtgängig, dass sich diese im Betrieb verschoben haben. Die Federnocken verhindern die Verschiebung.

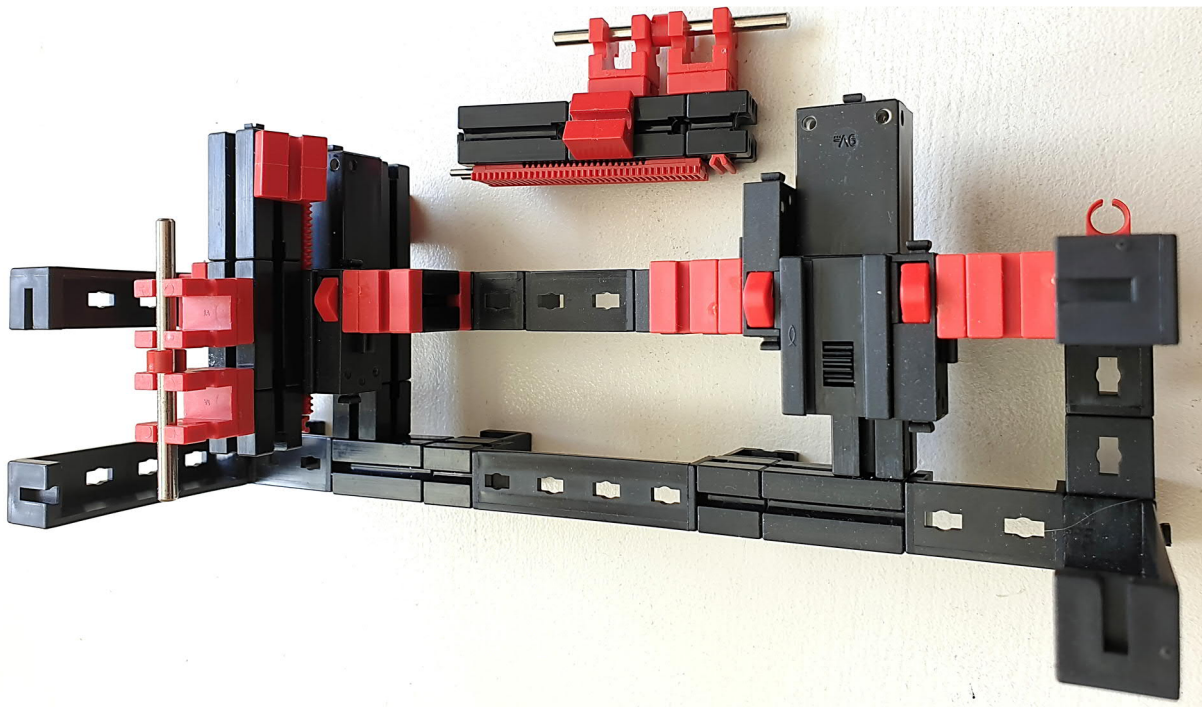


Abb. 13: Bauabschnitt für Stopper-Motoren aus der Frontalansicht

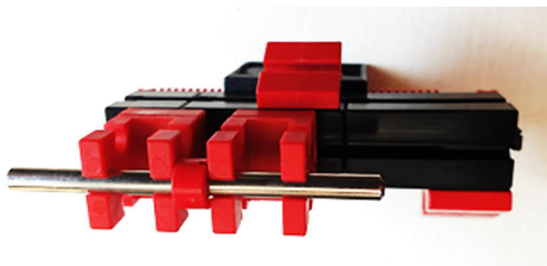


Abb. 14: Aufbau für die Zahnstangen von oben (links)



Abb. 16: Konstruktion für die Befestigung der Endtaster für die Stopper-Motoren

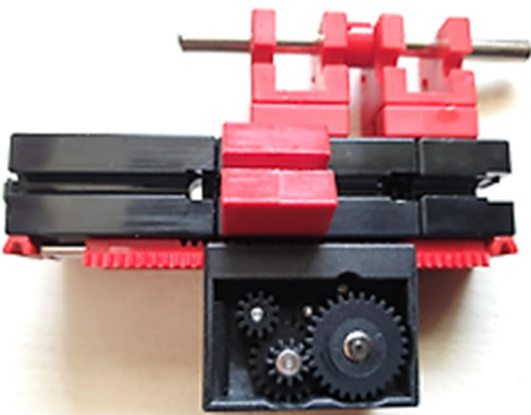


Abb. 15: Aufbau für die Zahnstangen von der Seite

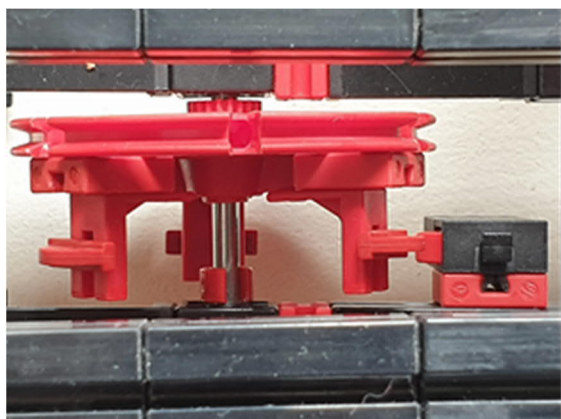


Abb. 17: Ersatzkonstruktion für die Schleifringe

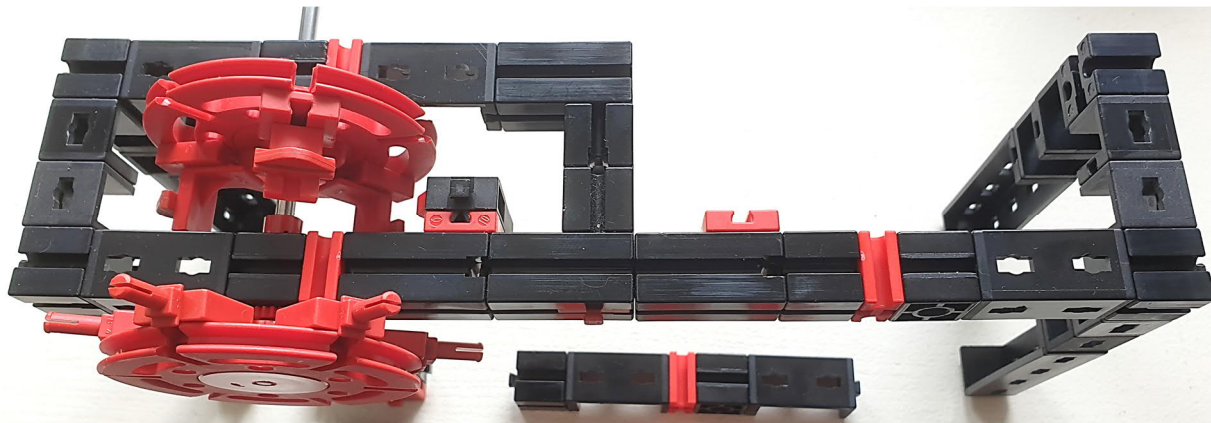


Abb. 18: Bauabschnitt für die Drehscheiben mit der Ersatzschleiferlösung

Aufsatz für die Drehscheiben

Dies ist der interessanteste Teil: Hier werden die Rutschkupplungen zusammen mit den Drehscheiben und der Ersatzkonstruktion für die Schleifringe aufgebaut.

Der Ersatzschleifring

In Abb. 17 ist die Ersatzkonstruktion für den Schleifring mit Taster zu sehen, in Abb. 20 die Noppen, die auf die Drehscheibe geschoben werden.

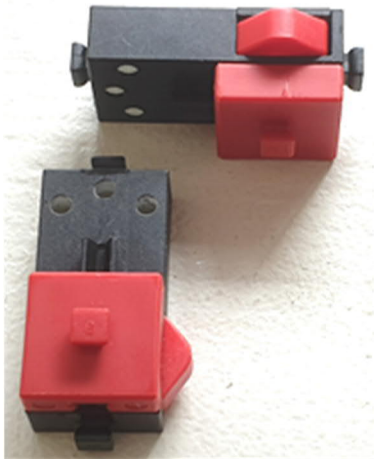


Abb. 19: Die Taster zur Feststellung, ob eine gerade Zahl angezeigt wird

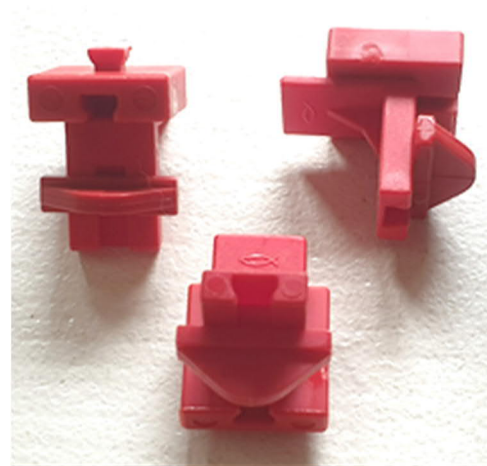


Abb. 20: Die Noppen an der Drehscheibe zur Betätigung der Taster

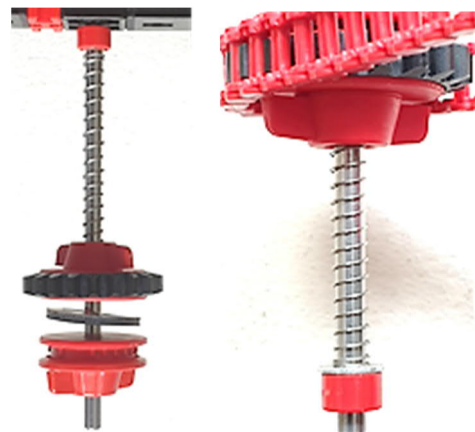


Abb. 21: Aufbau der Rutschkupplung (links); U-Scheibe vor der Klemmbuchse 5 mm (rechts)

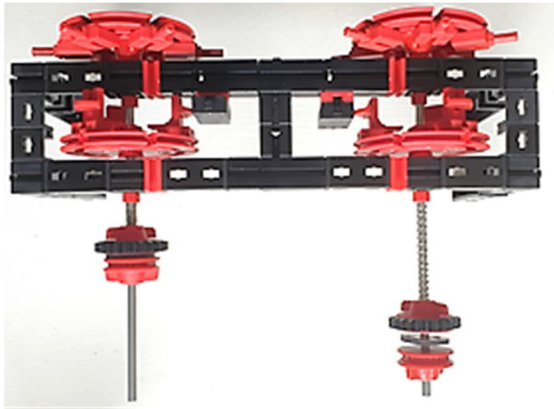


Abb. 22: Drehscheiben mit Ersatzschleifer und Rutschkupplung

Die Rutschkupplung

Die Federn aus dem Einzelteilesortiment ([132875](#)) sind scheinbar heute aus einem anderen Material oder haben eine andere Federkonstante. Bei meinen ersten Tests habe ich gemerkt, dass eine Feder vollkommen ausreichend ist; sie braucht auch kaum zusammengedrückt zu werden.

Auf der Rückseite kommt auf die Achse 170 zuerst eine Klemmbuchse, dann die U-

Scheibe, gefolgt von der Feder. Jetzt folgt das Zahnrad Z20 mit Flachnabe, wobei die Nabe vom Zahnrad zuge dreht wird, bis der Widerstand größer wird, sodass die Nabe nicht aufgeht, aber die Stange sich in der Nabe noch drehen lässt. Jetzt kommen der Dichtungsring und eine abschließende Flachnabe, die richtig zuge dreht wird (Abb. 21 und 22). Der richtige Abstand für die Feder muss im laufenden Betrieb der Drehscheiben ermittelt werden.

Aufsatz für den Antrieb

Den Abschluss bildet der Antrieb mit dem XM-Motor (M1) für die Drehscheiben. Zwischen den Z40 befindet sich eine Klemmbuchse 10, die die Zahnräder auf den richtigen Abstand bringt.

Hinweis: Die Bauteile für die Schaltscheibenkonstruktion aus der Original-Bauanleitung werden nicht benötigt und entfallen.

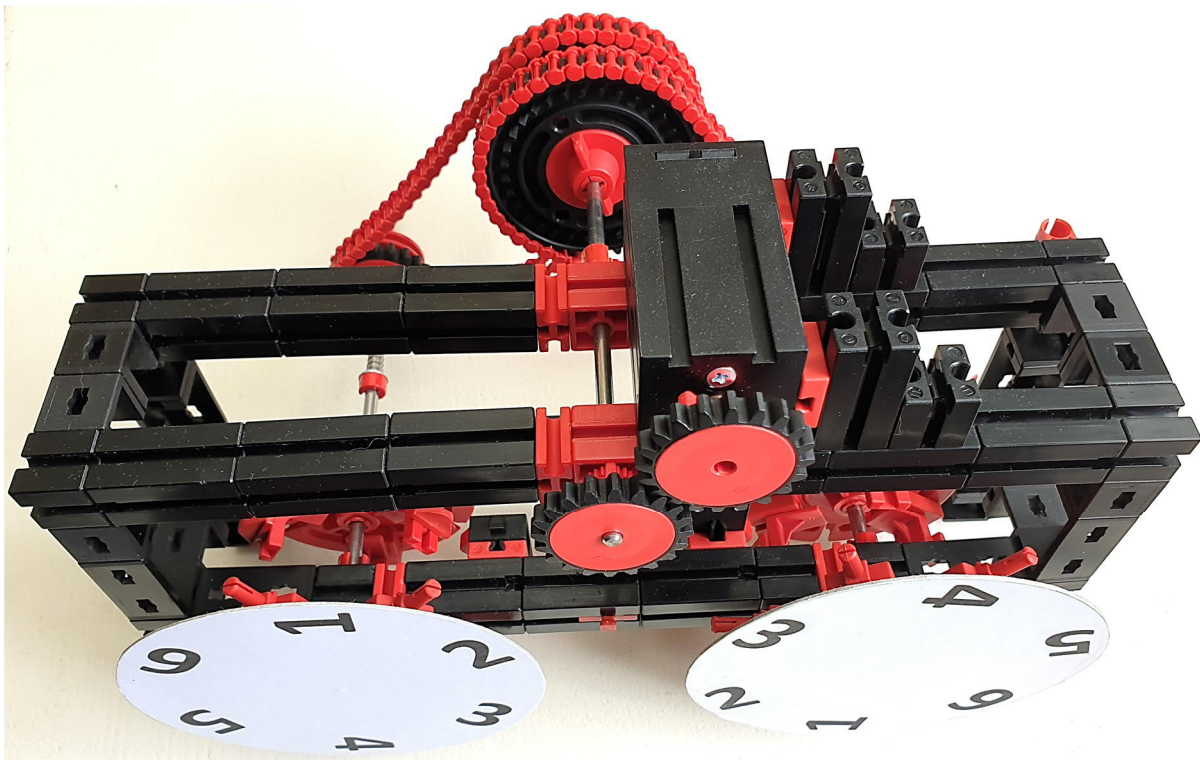


Abb. 23: Bauabschnitt für den Antrieb fertig montiert auf die Baugruppe für die Drehscheiben

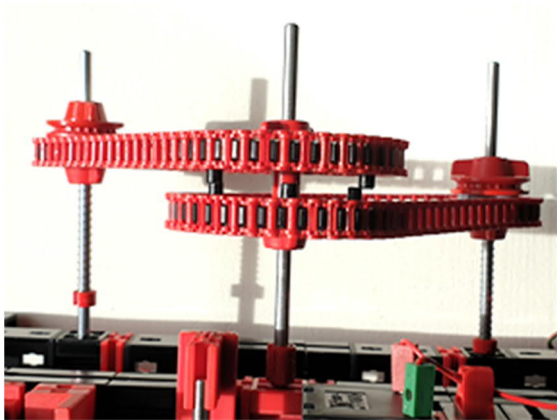


Abb. 24: Blick auf die fertig montierten Ketten von oben

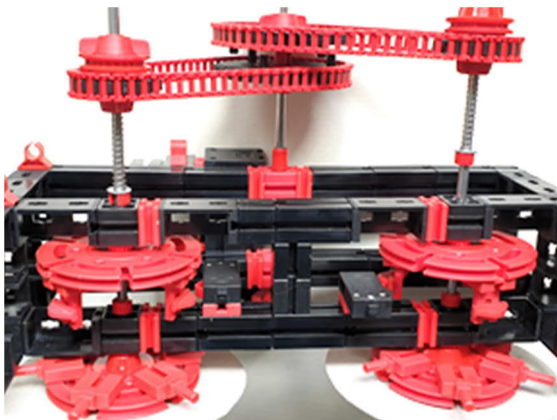


Abb. 25: Blick auf die fertig montierten Ketten von unten

Die Geldausgabe

Dieser Teil hat noch einmal ein paar Schwierigkeiten mit sich gebracht, weil die Konstruktion an einigen Stellen nicht gut dokumentiert bzw. das Bildmaterial von unzureichender Qualität war.

Die alte kleine rote Grundplatte lässt sich hervorragend durch die neue Grundplatte 120×60 ([35129](#)) ersetzen. Damit diese beim Einbau in den Spielautomaten auf die Winkelträger passt bzw. sich auf der gleichen Anbauhöhe befindet, muss noch ein Baustein 7,5 mit einer Federnocke auf der Unterseite der Grundplatte montiert werden.

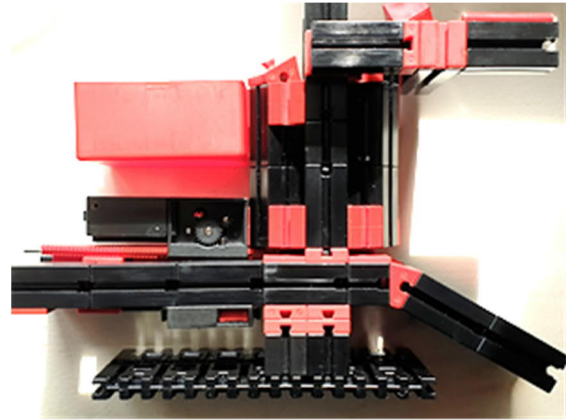


Abb. 26: Bauabschnitt für die Geldausgabeinheit (1)

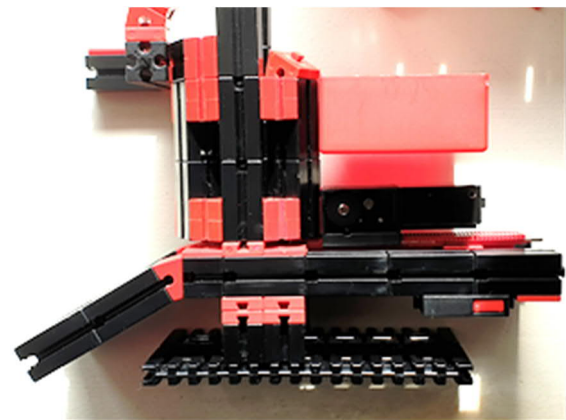


Abb. 27: Bauabschnitt für die Geldausgabeinheit (2)



Abb. 28: Nicht erkennbarer Teil in der Bauanleitung für das Club-Modell (1)

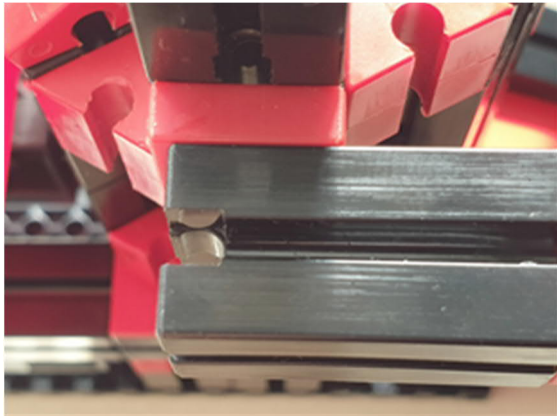


Abb. 29: Nicht erkennbarer Teil in der Bauanleitung für das Club-Modell (2)



Abb. 30: Montage der Rutsche mit dem Leuchtstein für den Geldeinwurf (1)

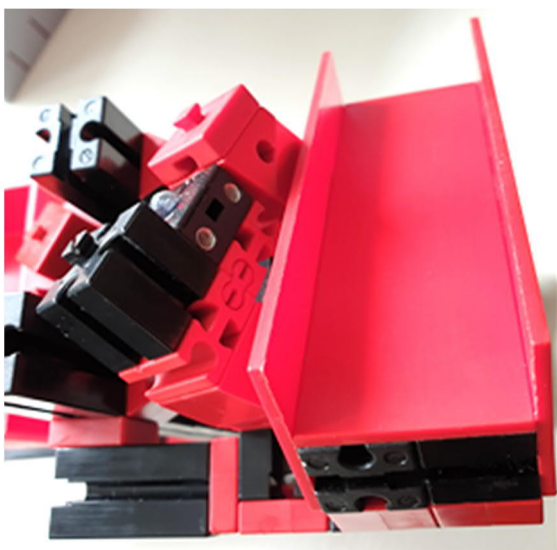


Abb. 31: Montage der Rutsche mit dem Leuchtstein für den Geldeinwurf (2)

In Abb. 29 sieht man den kleinen Versatz ganz deutlich, mit dem der Baustein angebracht werden muss. In Abb. 28 sind die Bausteine 5 oben solche mit zwei Zapfen.

Abb. 30 und 31 zeigen die Montage der Rutsche für das „Falschgeld“ und wie der LED-Baustein für die Lichtschranke vom Münzeinwurf montiert wird.



Abb. 32: Die Geldeinwurfschiene

Die Geldausgabe war damals noch für DM und Pfennige entworfen worden. Beim Test hat sich gezeigt, dass die Geldaufnahme für 1 € 20 Cent, 10 Cent und 5 Cent funktioniert; 2 € und 50 Cent passen nicht, während 1 und 2 Cent-Münzen durchfallen und über die seitliche Rutsche ausgeworfen werden – die Münzprüfung hat also noch Verbesserungsbedarf. Am besten geht es mit 1 €-Stücken, aber die 20 Cent sind fast genauso groß und dick, somit habe ich mich für 20 Cent als Zahlungsmittel für ein Spiel entschieden. Wer 1 € einwirft, bekommt trotzdem nur ein Spiel.

Problematisch ist das erste Geldstück im Sammelbehälter. Liegt erst einmal ein Geldstück drin, gibt es keine Probleme mehr. Aber 5 von 10 der ersten Geldstücke sind nicht liegen geblieben sondern durchgefallen. Auch bei der Ausgabe kam es hin

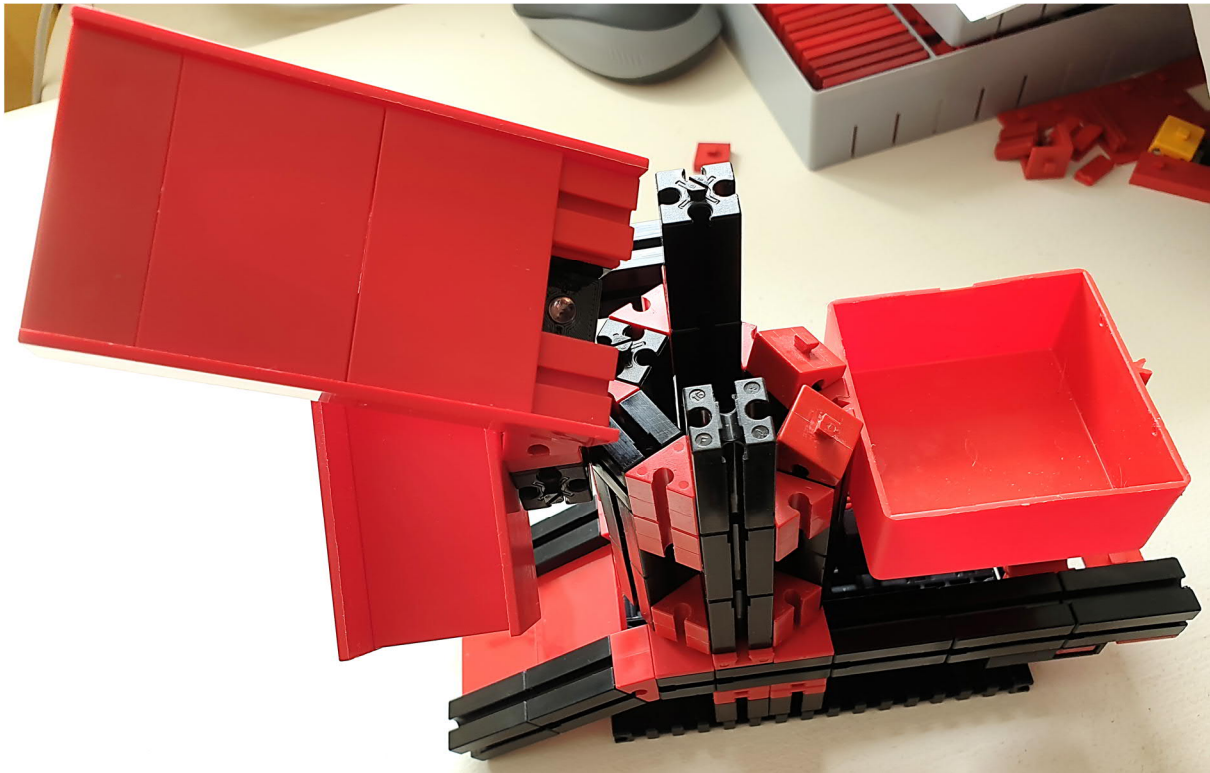


Abb. 33: Die fertige Geld-Ein- und -Ausgabeeinheit

und wieder dazu, dass sich etwas verklemmt hat, meistens beim Test mit 10 Cent-Stücken: Die sind einfach zu klein für die Konstruktion. In den Sammelbehälter passen 36 20-Cent-Münzen, bevor diese in den Auffangbehälter rutschen.

Justierung

Gut wäre, jede Baugruppe schon vor der Verkabelung separat auf Funktion zu prüfen (Stopper-Motoren, Drehscheiben, Geldausgabe) und zu justieren.

Bei der Justierung der Endstellungen für die Stopper-Motoren muss darauf geachtet werden, dass die Stange auf der Zahnstange nicht gegen die Drehscheibe fährt.

Bei der Justierung der Endstellungen für den Motor der Geldausgabe füllt man am besten ein paar Geldstücke in den Sammelbehälter ein. Vorne sollte die Spitze der Zahnstange zu sehen sein und hinten muss deutlich das Nachrutschen des Geldstapels zu hören sein, bevor der Motor stoppt.

Am schwierigsten war die Justierung der Rutschkupplung. Hier muss sich die zweite Drehscheibe auch noch drehen können, wenn die rechte, erste Drehscheibe gestoppt wurde. Nach einigen Versuchen und immer kleinen Korrekturen an der Einstellung des Federdrucks war die richtige Einstellung gefunden.

Ein kleiner Zusatz

Als kleinen Zusatz habe ich mir erlaubt, einen Rahmen (Abb. 34 und 35) hinzuzufügen, in dem die beiden ausschlaggebenden Gewinnzahlen zu sehen sind.

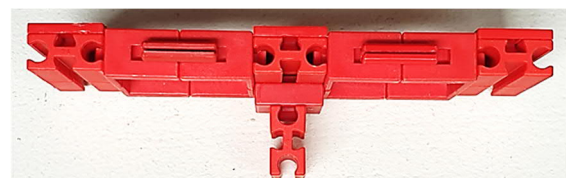


Abb. 34: Anzeigerahmen von oben

Der Anzeigerahmen wird zwischen den Drehscheiben mit einer Federnocke befestigt, die auf den Abb. 18 und 25 zu sehen ist.

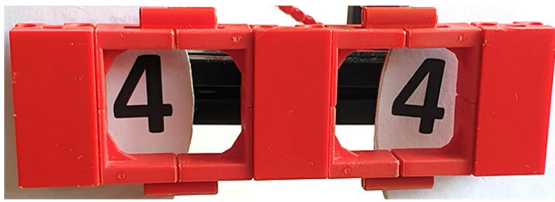


Abb. 35: Anzeigerahmen von vorne

Sonstige Anmerkungen

Der Fototransistor reagiert nicht auf LED-Licht; hier muss eine herkömmliche Lampe verwendet werden. Bitte darauf achten, dass „+“ an der rot markierten Stelle angeschlossen wird.

In der Praxis hat sich gezeigt, dass ein Multifunktionsmodul mit Schwellenwertregler nicht benötigt wird; das kann somit entfallen. Bei Verwendung eines Schaltnetzteils mit Power-Regler ist der Einsatz eines Versorgungsmoduls nicht erforderlich, nur bei Verwendung des Trafos mot.4 wird das Versorgungsmodul benötigt. Mit diesen Änderungen kann das Modell mit nur vier Elektronikmodulen aufgebaut werden.

Der Aufbau auf den Zahnstangen für die Geldausgabe hat das gleiche Problem wie bei den Stopper-Motoren: Er lässt sich sehr leicht verschieben und sitzt dann an der falschen Position. Hier habe ich den Baustein 5 durch einen Baustein 5 15×30 ([35049](#)) ersetzt. Das hat dahingehend Abhilfe geschaffen, dass der Aufbau nun nicht mehr so leicht seine Position verlässt (diese Änderung ist nicht in der Teileliste enthalten).

Bei Benutzung nur eines Schaltnetzteils mit einem Power-Regler ist es im Betrieb des Spielautomaten vorgekommen, dass die Mono-Flops geschaltet haben, wenn die Stopper-Motoren in Betrieb gegangen sind. Motoren erzeugen Abrissfunken und Spannungsspitzen, die die digitale Elektronik beeinflussen und auch über den Power-Regler weitergeleitet werden. Ein Kondensator mit 2200 µF z. B. aus dem Versorgungsmodul zwischen „+“ und „-“ nahe der

Elektronikmodule hat hier Abhilfe geschaffen.

Der letzte Hinweis gilt den Drehscheiben. Diese zentriert zu montieren und zusätzlich so, dass die Zahlen genau nebeneinanderstehen, war schon eine äußerst herausfordernde Aufgabe. Hier kann man kleine Korrekturen bei der Höhe der Zahlen vornehmen z. B. mit Hilfe einer Klemmbuchse 10 (oder Ähnlichem) auf der Achse des Stoppers.

Teileliste

In diesem Beitrag habe ich verschiedene Möglichkeiten und Alternativen vorgestellt. Die Teileliste (Tab. 1) enthält diejenigen Teile, die für den Aufbau des Spielautomaten in Abb. 36 benötigt werden.

Bauteil	Stk.	Art.-Nr.
Grundplatte 120×90	3	182622
Baustein 5	39	37237
Federnocken	8	31982
Baustein 7,5	42	37468
Baustein 15 sw	25	32881
Baustein 15 mit Bohrung sw	4	32321
Winkelstein 10×15×15	20	38423
Winkelstein 30°	14	31011
Achse 60	2	31032
Klemmbuchse 5	6	37679
Nabenmutter	12	31058
Zahnrad Z20 sw	4	36334
Verbindungsstück 15	8	31060
Verbindungsstück 45	2	31330
Radachse	12	36586
Kassette	1	130961
Bauplatte 15×15	4	38246
Bauplatte 15×45	3	38242
Bauplatte 15×90	2	38245

Bauteil	Stk.	Art.-Nr.
Bauplatte 30×90	3	38251
U-Scheibe 4 mm	2	161261
Druckfeder 0,3×5×45	2	132875
Dichtungsring ½"	2	-
Motor XS	3	137096
Zahnstangengetriebe sw	3	37272
Zahnstange 30	1	37457
Verteilerplatine	4	185518
Fototransistor	1	36134
Leuchtstein sw	1	38216
Mono-Flop-Modul	2	70005
Stecker grün	49	181583
Power Regler	1	139778
Kabelhalter	20	35969
Grundplatte 120×60	1	35129
Baustein 5 mit 2 Zapfen	4	37238
Kufe	6	31602
Baustein 15 mit 2 Zapf. sw	11	32882
Baustein 30 sw	64	32879
Baustein 15 mit Bohrung rt	2	32064
Drehscheibe 60	4	31019
Winkelstein 60°	14	31010
Achse 170	3	35696
Klemmbuchse 10	2	31023
Flachnabenzange	12	35031
Zahnrad Z40 sw	2	31022
Verbindungsstück 30	1	31061
Riegelscheibe	5	36334
Rollenlager	4	37636
Kettenglieder (2 × 67)	144	36248
Bauplatte 15×30	10	38241
Bauplatte 15×60	1	38464

Bauteil	Stk.	Art.-Nr.
Bauplatte 30×45	2	38248
Zahlen zum Aufkleben	2	143389
Flachstein 30	2	31013
U-Scheibe im Dichtungsring	2	-
Scheiben mit Zahlen 100 mm	2	-
Motor XM	1	135485
Mini-Taster	8	37783
Zahnstange 60	3	37351
Gehäuse Verteilerplatte	4	152059
Störlichtkappe 6 mm	1	36532
Linienstecklampe	1	37875
Relaismodul	2	70018
Stecker rot	45	181584
Schaltnetzteil	1	505287

Tab. 1: Teileliste für den Spielautomaten in Abbildung 36

Die Anzahl der Kabelhalter kann als Richtwert genommen werden, weil jeder die Kabel nach seiner Art verlegen wird.

Der Spielautomat ist fertig

Letzter Schritt ist die Feinabstimmung für die Zeitintervalle von Stopper rechts (Motor M3), Stopper links (Motor M2) und Antrieb (Motor M1) aufeinander.

Jetzt bleibt mir nur noch viel Spaß beim Bauen und Tüfteln zu wünschen.

Quellen

- [1] santjohanser: [Spielen. Lernen. Technik](#). Autorisierter fischertechnik-Fachhändler.
- [2] fischertechnik: *Club-Modell 3/79 Bauanleitung »Spielautomat«*. Auf docs.fischertechnikclub.nl.

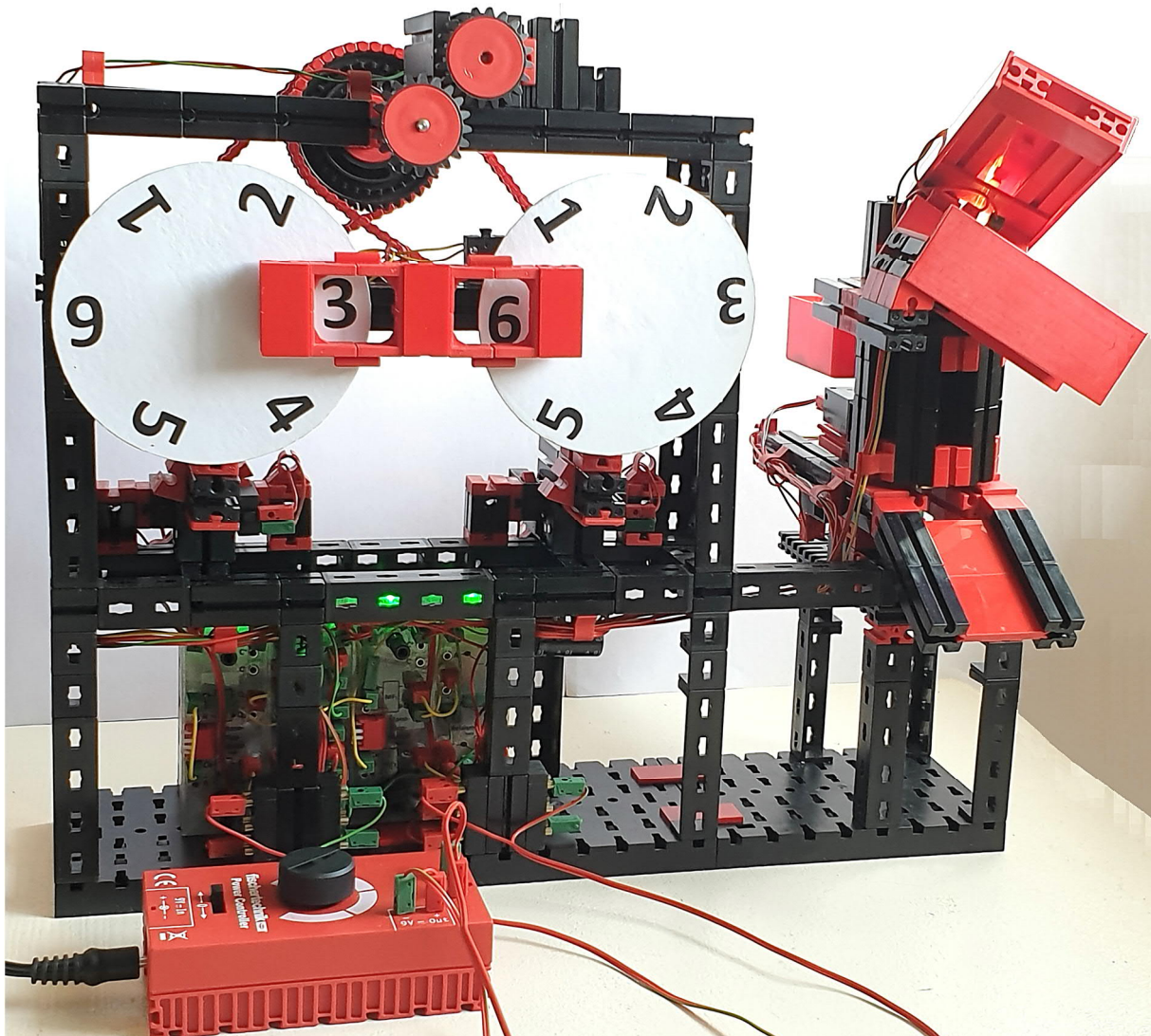


Abb. 36: Der fertig aufgebaute Spielautomat

- [3] Hans-Christian Funke: *Beschriftung Drehscheiben 100mm*. Zum Herunterladen und Ausdrucken im [Downloadbereich dieser Ausgabe](#) der ft:pedia.

Elektronik

Die Praktika von fischertechnik im Jahr 2022

Arnoud van Delden

Mit dem „hobbylabor 1“ ([30626](#) & [30850](#)), dem „Elektronik-Praktikum“ ([30629](#)) und dem „IC-Digital-Praktikum“ ([30630](#)) stellte fischertechnik in den 1970er Jahren mehrere Experimentierkästen mit diskreten elektronischen Bauelementen und digitalen ICs vor. Da komplette Praktika- und Hobbylabor-Baukästen aufgrund ihres hohen Sammlerwerts heute kaum noch zu finden sind und es fast unanständig erscheint, mit diesen Museumsstücken tatsächlich zu spielen, habe ich mir eine Lösung einfallen lassen, in der sich alle Praktika-Experimente der Vergangenheit auf modernere Art wiederfinden. Bleibt die Ära der diskreten Elektronik und der digitalen ICs nicht das Fundament, auf dem die heutigen Mikrocontroller aufgebaut sind?

Verschiedene Konstruktionsmöglichkeiten

Beim hobbylabor 1, erschienen 1974, behandelte das dicke Begleitbuch [1] die allgemeine Theorie der Elektronik und des Magnetismus. Der Zusatz „1“ deutete an, dass fischertechnik zunächst einen Folgebaukasten geplant haben könnte. Er erschien jedoch nicht. Drei Jahre später, im Jahr 1977, wurden zwei Praktika-Boxen gleichzeitig veröffentlicht. Diese machten mit einem eher praxisnahen Ansatz, bei dem verschiedene Modelle mit der verbauten Elektronik und Digitaltechnik gesteuert wurden, ihrem Namen alle Ehre.

Auffallend war, dass Kondensatoren, Widerstände, Transistoren und digitale ICs in ihrer verfügbaren regulären Gestalt verwendet wurden. Die typischen rot-grauen Konstruktionselemente fehlten komplett; den Praktikumsheften wurde hinten sogar eine Stückliste beigelegt, um eventuell defekte Teile nachkaufen zu können.

Alles in allem hatte jede Box ihr eigenes „Montagesystem“ um die Schaltungen aus Komponenten aufzubauen (Abb. 1). Beim Hobbylabor waren das Modul „EF

Experimentierfeld“ ([37140](#)) und das Modul „PB Doppelpoti“ ([37158](#)) noch formkompatibel zu den Gehäusen der Silberlinge. Aber das „Volt-Amperemeter“ ([37142](#)) hatte seltsamerweise ein Gehäuse mit den gleichen Abmessungen, aber zwei Befestigungsfedern (statt einer) an den kurzen Seiten. Zum Inhalt gehörten auch einige kleine Kunststoffmodule, mit denen Verbindungen hergestellt und Bauteile auf dem Experimentierfeld montiert werden konnten (Abb. 1).

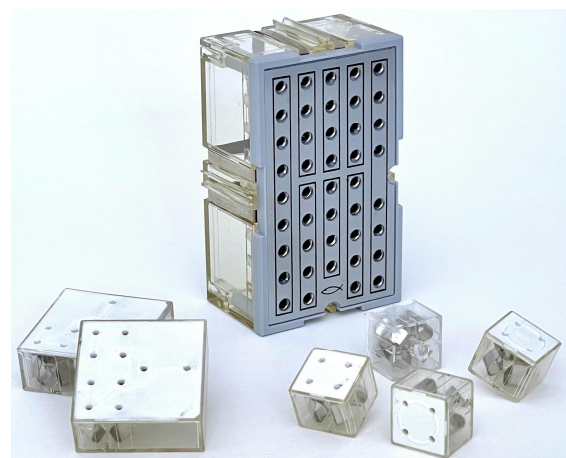


Abb. 1: Komponenten des hobbylabor 1 und des Elektronik-Praktikum

Die 1977 erschienenen Praktikums-Boxen enthielten mit den ICs Teile, die sich aufgrund des Abstands der Anschlussdrähte nicht mit Steckern auf dem Experimentierfeld befestigen ließen. Die Funktion des „Experimentierfeldes“ des Hobbylabors übernahmen beim Elektronik-Praktikum [2] zehn kleine Module aus durchsichtigem Kunststoff, mit denen die kleinen Schaltungen aufgebaut werden konnten. Auf Wunsch ließen sich damit auch Schalter und Verbindungen herstellen. Für die Verbindung der beiden Transistoren in der Box waren zwei größere Module vorgesehen.

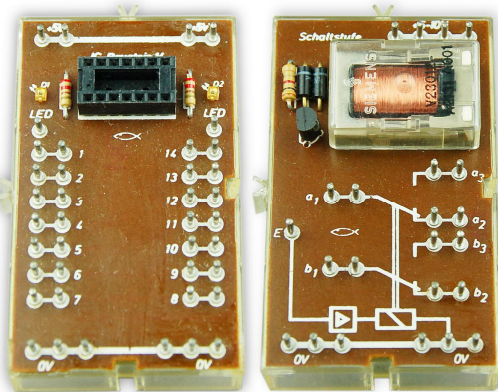


Abb. 2: IC-Fassung und Schaltstufe des IC-Digital-Praktikums

Das IC-Digital-Praktikum [3] enthielt ähnlich große Module wie die Silberlinge, jedoch mit einer anderen Höhe (Abb. 2). Der „Baustein Schaltstufe“ (38617) enthält ein Relais mit Vorverstärker. Die beiliegenden ICs (NAND 7400, NOR 7402, AND 7408 und Monoflop 74121) konnten im Fuß der „Baustein IC-Fassung“ (38616) platziert werden; danach ließen sich die Anschlüsse mit Leiterplattenpins verbinden. Eine Abschirmung mit einer schematischen Darstellung des verwendeten ICs konnte als Referenz zwischen den Reihen von Verbindungsstiften platziert werden. An der Ober- und Unterseite der Platine dieser Module konnten jeweils die Versorgungsspannung und 0 Volt durchgeschleift werden.

Ein „Steckbrett-Silberling“

Wie eingangs erwähnt bildet das Gebiet der diskreten Elektronik die Grundlage aller modernen Mikroprozessorentwicklungen. Zwar kannte ich die Konstruktionshefte im digitalen PDF-Format, doch in den letzten Jahren habe ich auch die Papierausgaben in niederländischer Sprache erhalten. Die vielfältigen Experimente in diesen klassischen fischertechnik-Boxen regen nach wie vor die Fantasie an und so entstand die Idee, die Hefte mit den ursprünglich vorgeschlagenen Bauteilen und dazugehörigen Modellen systematisch durchzuarbeiten.

Außerdem suchte ich für meinen „normalen“ Elektronikarbeitsplatz schon länger nach einer Möglichkeit, beim Testen einer Schaltung Messleitungen und Prüfspitzen stabil und übersichtlich anbringen zu können (Abb. 3, 4).

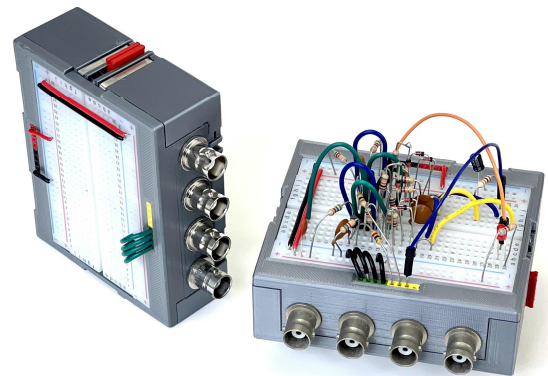


Abb. 3: Silberling XL-Gehäuse mit Steckbrett und Messausgängen

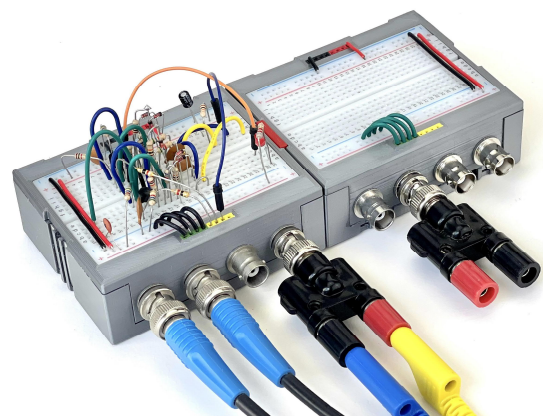


Abb. 4: Anschluss symmetrischer und asymmetrischer Signale

Für den Zauberling [4] hatte ich ein eigenes Silberling-Gehäuse entworfen, auf dem natürlich diverse 3D-gedruckte Anschluss-Variationen möglich sind. Ich habe zuvor eine Variante für ein universelles Stromversorgungsmodul entworfen, das neben den herkömmlichen 9 Volt verschiedene andere Spannungen liefern kann [5].

Als modernes Äquivalent zu den verschiedenen Aufbaumöglichkeiten könnte das gängige „Steckbrett“ dienen. Tatsächlich lassen sich damit alle Möglichkeiten des „Experimentierfeldes“, die Module aus dem IC-Digital-Praktikum und die diversen anderen vorgeschlagenen klassischen Montagemöglichkeiten abdecken. Neben den konventionellen Bauteilen lassen sich auch digitale ICs einfach montieren.

Die Steckbretter sind in verschiedenen Größen erhältlich. Vor allem ein kleines Modell von 8,5 · 5,5 cm schien mir sehr brauchbar und hatte die ideale Größe, um in einem „Silberling XL“-Modul mit doppelter Breite untergebracht zu werden.

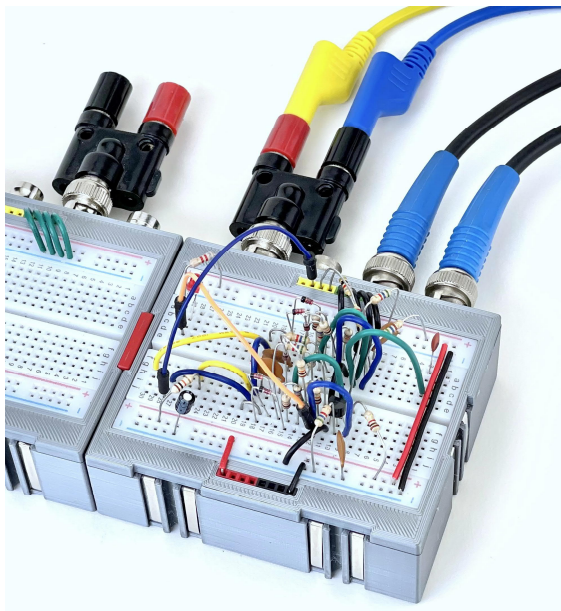


Abb. 5: Einfache und übersichtliche Messung an Schaltungen

Mittlerweile verfügt mein Elektroniklabor über Messgeräte, von denen ich nur träumen konnte, als fischertechnik die hier besprochenen Baukästen auf den Markt

brachte. Ein Modell mit Messbuchsen auf der Rückseite ermöglicht ein einfaches Messen. So können BNC-Kabel und, über einen Adapterstecker, auch 4-mm-Bananenstecker fest angeschlossen werden (Abb. 5).

Die Buchsen auf der Rückseite werden im Inneren des Gehäuses zu Steckerleisten geführt, in denen die gleichen Kabel wie auf dem Steckbrett verwendet werden können. Auch die 9-Volt-Versorgungsspannung kann so der Schaltung auf der Platine zugeführt werden.

Manchmal ist es angenehm, auch 5 Volt zur Verfügung zu haben. Ich habe mir deshalb ein Modul gebastelt, das diese Versorgungsspannung zusätzlich zu den 9 Volt erzeugt, die durch die Seiten der Silberlinge durchgereicht werden, und auf einer separaten Verteilerleiste zur Verfügung stellt (Abb. 6). Dies ist praktisch für die TTL-ICs, die im IC-Digital-Praktikum verwendet werden, oder für zukünftige Experimente mit Mikrocontrollern.

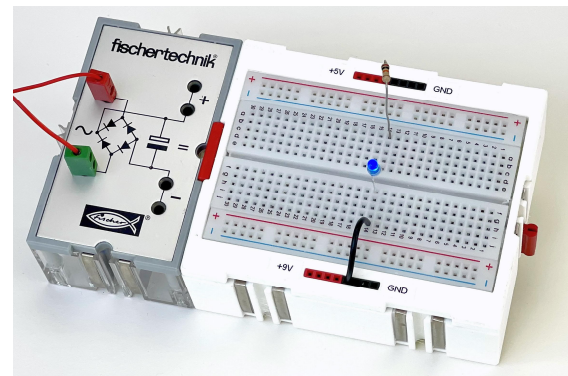


Abb. 6: Der „Steckbrett-Silberling“ mit Verteilerleiste für 5 und 9V

Der „Peripherie-Silberling“

Der „Steckbrett-Silberling“ hatte sich bereits einen Platz in meinem „Labor“ – der Elektronikwerkstatt – erobert. Wird mehr Platz benötigt als das relativ kleine Steckbrett bietet, lassen sich durch den modularen Aufbau problemlos mehrere „Steckbrett-Silberlinge“ zusammenschieben. In der Praxis wird jedoch oft viel wertvoller

Platz auf dem Steckbrett durch Schalter, Taster oder ein benötigtes Potentiometer verbraucht. Diese zusätzlichen Teile sind in der Regel schwierig auf einem Steckbrett zu montieren und nehmen unverhältnismäßig viel Platz ein. LEDs lassen sich meist einfach stecken; wer damit aber mehrere Signale überwachen möchte, braucht schnell viel Platz für die diversen Vorwiderstände.

So wünschte ich mir schon vor einer ganzen Weile eine universelle Leiterplatte für diese immer wiederkehrende „Peripherie“. Auf der Wunschliste standen als Mindestanforderung ein paar Druckknöpfe, Schiebeschalter und einige LEDs, die als universelle Anzeige von Signalen aus der zu testenden oder zu entwerfenden Schaltung dienen könnten. Allerdings habe ich online keine derartigen, geeigneten Produkte gefunden.

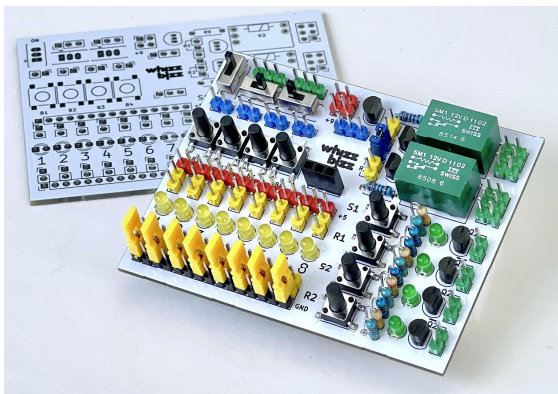


Abb. 7: Viel nützliche Peripherie auf einer Leiterplatte

Daneben entstand der Wunsch, mit dem Steckbrett-Silberling die klassischen Experimente der Praktikumskästen durchführen zu können. Und obwohl der „RB1 Relaisbaustein“ ([36392](#)) bei Bedarf natürlich mit dem 3D-gedruckten Steckbrett-Silberling-Modul verbunden werden kann, wünschte ich mir für einige Versuche ein kleines Relais. Ein Vorteil eines klassischen Relais ist schließlich, dass sich auch die Drehrichtung eines Motors einfach ändern lässt – wie

beim Schaltungsbeispiel von „Ein uner müdlicher Schrägaufzug“ auf Seite 33 des Buchs zum IC-Digital-Praktikum [3].

Schließlich suchte ich nach einer einfachen Möglichkeit, „saubere“ Taktimpulse für Experimente mit getakteten digitalen ICs und (fischertechnik-) Flip-Flops zu machen. Wer schon einmal versucht hat, den „Flip-Flop FF-Baustein“ ([30815](#)) mit einem Taster am Eingang „CP“ zu steuern, weiß, dass das Prellen im Schaltmoment des Tasters als mehrere Taktimpulse interpretiert werden kann. Dieses Signal kann oft mittels eines Entstörkondensators (100nF) ausreichend entprellt werden, aber diese Lösung ist sicher nicht optimal.

Als Erweiterung des Steckbrett-Silberlings habe ich daher ein kleines Modul im gleichen „Silberling XL“-Gehäuse konstruiert, das acht LEDs, ein Potentiometer, zwei Schiebeschalter, vier Taster, zwei Relais und zwei „SR-Latches“ ohne Kontaktgeräusche beherbergt (Abb. 7). Das Modul verfügt über einen eigenen Spannungsregler für die 5-Volt-Stromversorgung und kann mit einem einfachen 9-Volt-Gleichstrom-Adapter, der an der Rückseite eingesteckt wird, mit Strom versorgt werden. Auch der klassische „GB Gleichrichter Baustein“ ([30811](#)) kann als Stromquelle verwendet werden (Abb. 8).

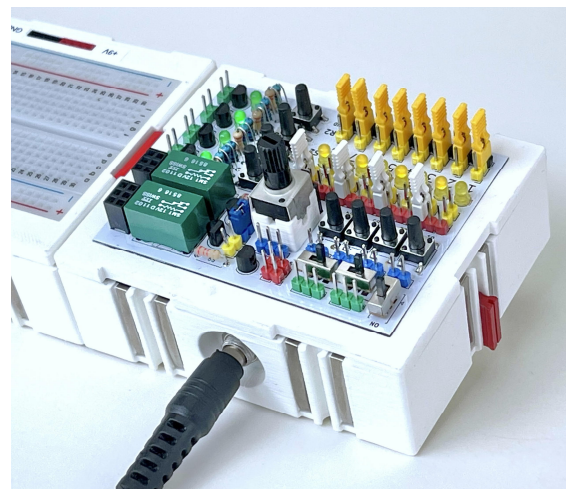


Abb. 8: Es können sowohl ein DC-Adapter als auch das gängige GB-Gleichrichtermodule als Stromversorgung verwendet werden

Die LED sind auf der Rückseite der Platine mit Vorwiderständen bestückt und können mittels Jumper einseitig mit den Plus- oder den Masse-Steckerleisten verbunden werden. Dadurch können sie sowohl als Indikator auf der „Open-Collector“-Seite (z. B. bei der Überwachung von Bussen) als auch für Signale mit „high-aktivem“ Potenzialpegel verwendet werden. Wird bei einer LED gar kein Jumper gesetzt, kann diese bei Bedarf sogar auf einer völlig unabhängigen Potentialebene verwendet werden. Das bietet viel mehr Flexibilität als die meisten vorgefertigten Drucktasten- und LED-Platinen, die online erhältlich sind. Die Auswahl entweder für eine „gemeinsame Anode“ oder eine „gemeinsame Kathode“ ist für diese Hilfsleiterplatten normalerweise zu restriktiv.

Die SR-Latches können mit ihren Druckknöpfen „S“ und „R“ abwechselnd gesetzt oder zurückgesetzt werden. Der HIGH-Ausgang (TTL-Pegel, positive Logik) wird durch eine leuchtende LED angezeigt. Außerdem ist der komplementäre Ausgang immer verfügbar.

Das Original-Hobbylabor-Modul „PB Doppelpoti“ ([37158](#)) enthält zwei Potentiometer von 1 k Ω bzw. 10 k Ω . Beide Potentiometer sind mit einem Vorwiderstand am Schleifkontakt gegen zu hohe Ströme geschützt. Das 10 k Ω -Potentiometer hat einen internen Schutzwiderstand von 1 k Ω , beim 1 k Ω -Potentiometer beträgt dieser Wert 150 Ω .

Ich dachte, der Schutzwiderstand hätte wenig Priorität, schließlich könnte er bei Bedarf ganz einfach auf dem Steckbrett platziert werden. In der Praxis schien mir angesichts der Experimente in den Büchern eine einzige Position für ein Potentiometer ausreichend. Um jedoch den für ein Projekt am besten geeigneten Potentiometerwert auswählen zu können, habe ich austauschbare Module gedruckt, mit denen ein Potentiometer von 4k5 Ω , 10 k Ω , 45 k Ω oder 500 k Ω auf der Platine platziert werden kann.



Abb. 9: Modular austauschbare Potentiometer

Bei den Relais entschied ich mich für zwei separate mit je einem Wechsler, da dies zusätzliche Flexibilität gegenüber einem Relais mit zwei Wechslern bietet: Mit einem Jumper kann der verstärkte Signaleingang beider Relais zu einer Relaisfunktion mit doppeltem Wechslerkontakt verknüpft werden. Natürlich muss dann nur ein Schaltsignal vorgesehen werden.

Das Board bietet zusätzliche Anschlüsse für direkte Verbindungen der Testschaltung auf dem Experimentierbrett mit Masse, 5 und 9 Volt.

Ein universelles Werkzeug

Die Grundlage zum Studium und Nachbau der Experimente und Modelle aus den historischen Baukästen schien mir so technisch ausreichend gelegt. Noch bevor sie ihre Daseinsberechtigung in Kombination mit fischertechnik bewiesen hatten, haben sich die vorgestellten Module bereits in verschiedenen Kombinationen bei Tests und Messungen in diversen Elektronikexperimenten bewährt (Abb. 9).

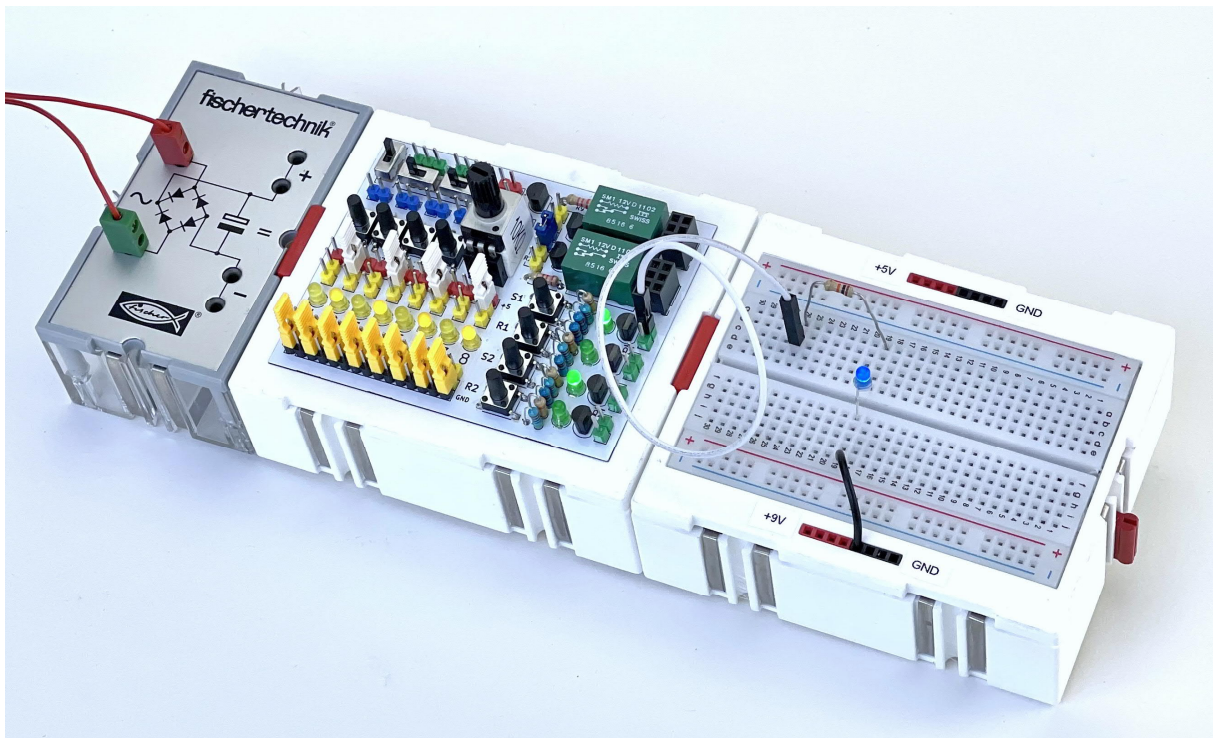


Abb. 10: Universell „Silberling-kompatibel“ – Aufbau für Experimente

Messleitungen sind nicht mehr im Weg und lösen sich nicht mehr. Der „Peripherie-Silberling“ und der „Steckbrett-Silberling“ bieten ausreichend Platz, um Prinzipschaltungen aufzubauen. Durch die austauschbaren Potentiometer ist der am besten geeignete Wert schnell ausgewählt. Die SR-Latches ermöglichen, „saubere“ Impulspegel ohne Prellen zu erzeugen. Das ist praktisch, wenn man mit CMOS-Zählern und getakteter Logik wie Flip-Flops experimentiert.

Universell einsetzbar

Bevor ich die Experimente aus den fischertechnik-Baukästen tatsächlich beginnen konnte, hatte sich der Steckbrett-Silberling bereits bei der Untersuchung des fischertechnik „Flip-Flop FF-Bausteins“ bewährt:

Nach einer Diskussion im ftCommunity-Forum [6] suchte ich nach einer einfachen Möglichkeit, die ursprüngliche Schaltung zu studieren. Mein Original-Fischertechnik-Flip-Flop erzeugte beim Einschalten (Anlegen der Versorgungsspannung mit dem

Original-418-Übertrager über das Gleichrichtermodul) nicht das erwartete LOW-Ausgangssignal. Da ich den Einfluss des großen Kondensators im Gleichrichtermodul und der Signallampe untersuchen wollte, habe ich die Schaltung auf dem Steckbrett aus separaten Bauteilen aufgebaut, um mit Bauteilvarianten einfach experimentieren zu können. Damit fand ich schließlich eine Lösung, die keine Änderungen innerhalb des ursprünglichen Silberlings erforderte.

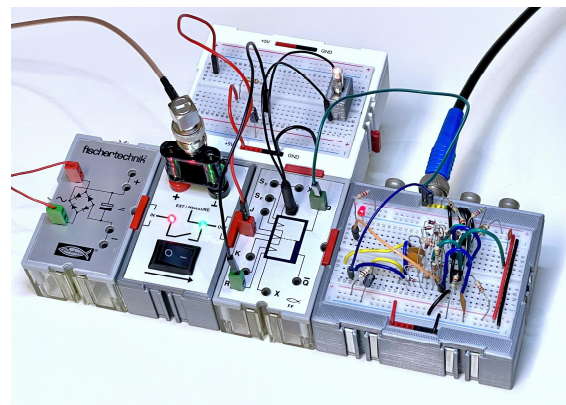


Abb. 11: Flip-Flop-Messungen – Versorgungsspannung schalten und testen

Weil ein zusätzliches 3D-gedrucktes Gehäuse wenig Aufwand verursacht, habe ich auch gleich ein Zwischenmodul gebastelt, um die Versorgungsspannung beliebig ein- oder ausschalten zu können, ohne an den roten Verbindungssteckern herumzufummeln. Damit ist es auch möglich, Messungen an der regulären Versorgungsspannung durchzuführen oder an dieser Stelle selbst Spannungen einzuspeisen. Das macht das Messen von Silberlingen unter Laborbedingungen etwas übersichtlicher.

Mehrere Möglichkeiten der Steuerung

Ein Modell, das sowohl im Buch des Elektronik-Praktikums [1, S. 76] als auch im IC-Digital-Praktikum [2, S. 86] auftaucht, ist das „Langzeit-Schrittschaltwerk“ (oder die „Universelle Schaltuhr“). Es ist ein interessantes Modell, weil seine Steuerung sowohl mit diskreten Komponenten als auch mit digitalen ICs und einem Relais gelöst wird.

Beim Elektronik-Praktikum wird mit dem Transistor BC238 ein per Knopfdruck getriggertes 10-Sekunden-Timer realisiert. Nach Ablauf dieser Zeit wird der BD135 als Motortreiber geöffnet, woraufhin eine Scheibe mit Nocken eine Sechstelumdrehung weiterdreht. Die Scheibe macht also pro Minute eine Umdrehung. Durch passende Übersetzungen kann dann beispielsweise eine Lampe etwa einmal pro Stunde geschaltet werden. Das Getriebe verlangsamt so sehr, dass die letzte Scheibe vierzig Stunden brauchen würde, um sich einmal zu drehen. Das erinnert fast an eine Ewigkeitsmaschine [7, 8].

Ich habe die Schaltung mit einigen Originalbauteilen und den Schaltelementen aus dem Original-Elektronik-Praktikum aufgebaut (Abb. 12). Im IC-Digital-Praktikum wird das gleiche mechanische Modell verwendet und die gleiche Funktionalität mit einem 74121 Monoflop IC als Timer und einem Relais zum Schalten des Motors umgesetzt.

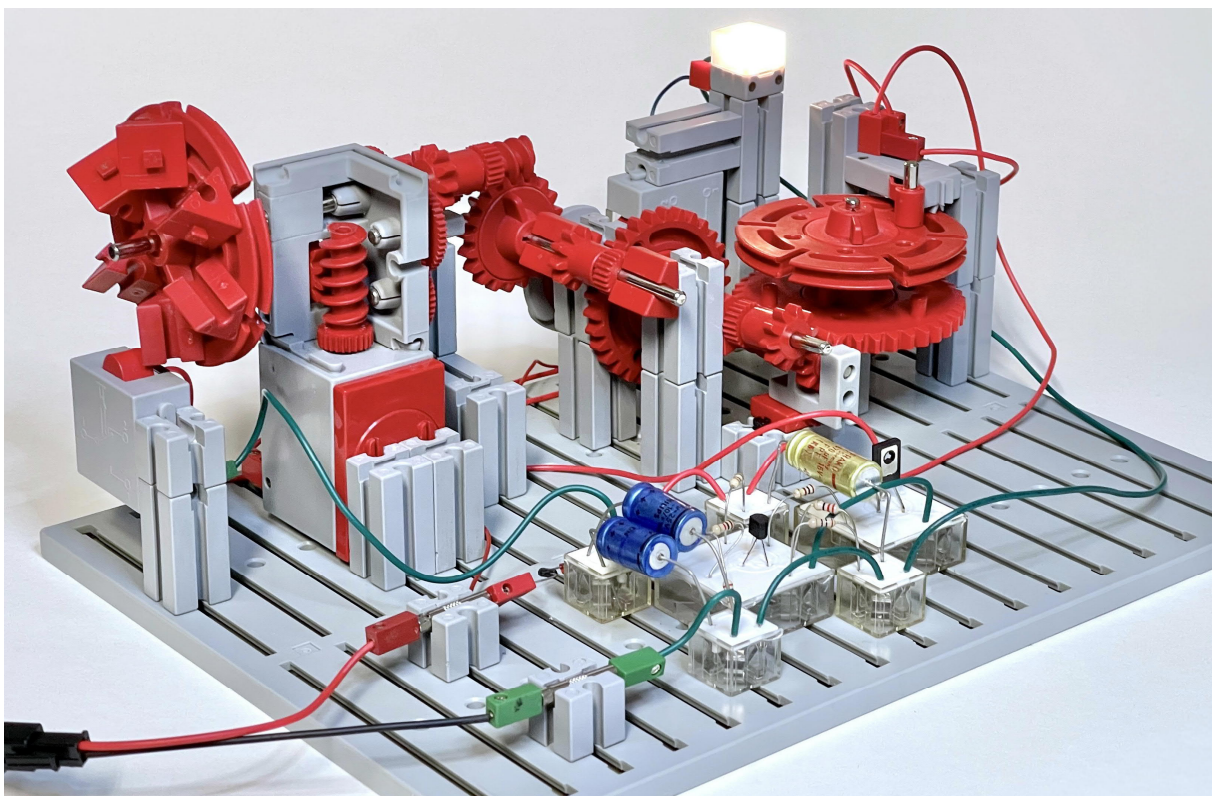


Abb. 12: Elektromechanischer Zeitschalter aus diskreten elektronischen Komponenten

Es macht Spaß und ist lehrreich, beide Lösungen nebeneinander auf einem Steckbrett-Silberling zu sehen. Der Messausgang ermöglichte die Untersuchung des Entladezyklus des Kondensators. Um jedoch beide Zeitschaltkreise abwechselnd zu testen, muss der Impulsscheibenschalter umgepolt werden. In Abb. 13 ist der Transistor-Timer aus dem Elektronik-Praktikum aktiv.

Der blaue Taster ersetzt hier den Schalter auf dem „Minutenrad“ der Lösung aus dem IC-Digital-Praktikum. Beide Schaltungen können abwechselnd mit dem gleichen Ergebnis verwendet werden. Sie machen eigentlich genau dasselbe, sind aber jeweils anders elektronisch aufgebaut. In diesem [Video](#) sieht man das Modell in Aktion.

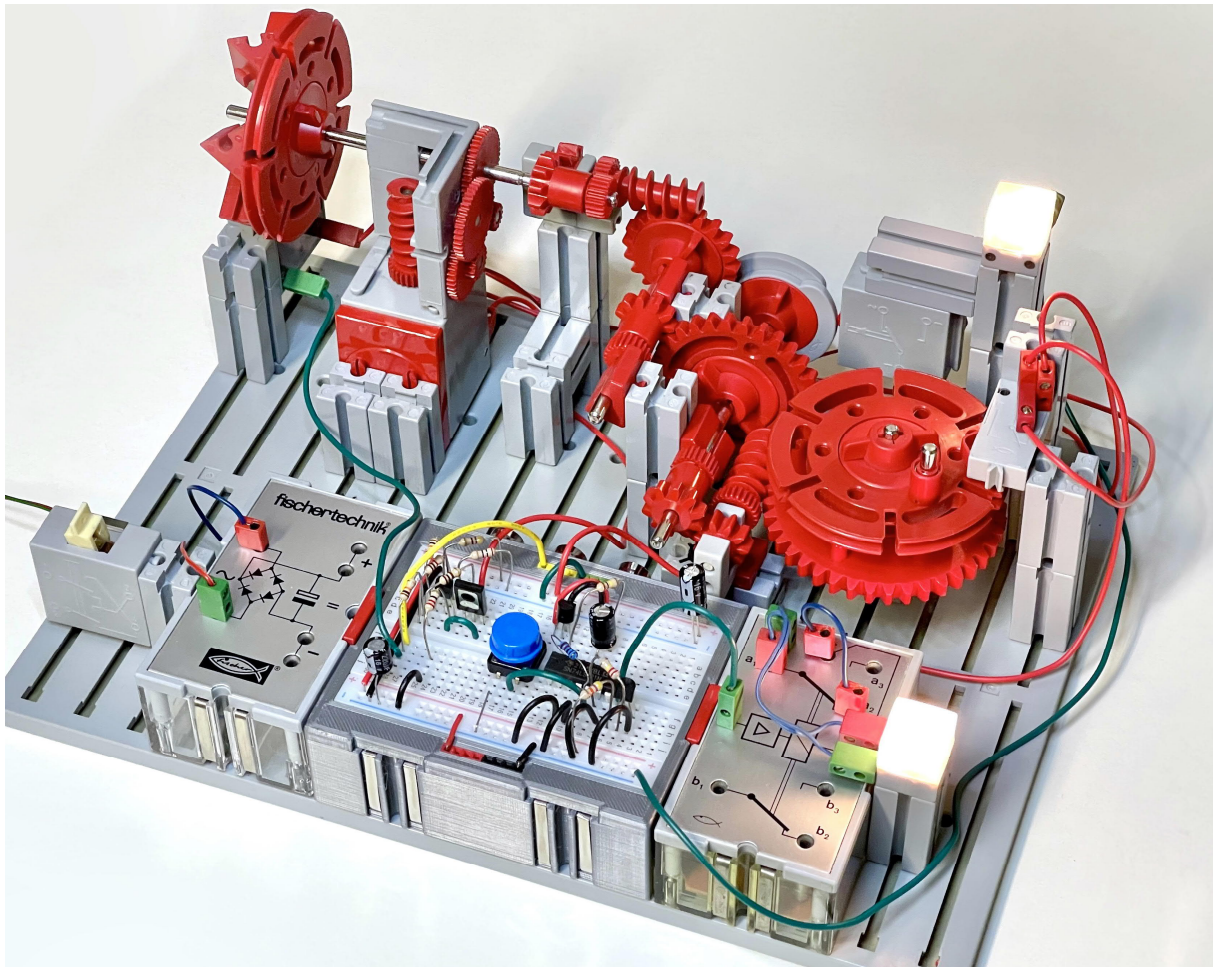


Abb. 13: Derselbe Timer mit zwei unterschiedlichen elektronischen Steuerungen auf dem Steckbrett-Silberling

Fazit

Das hobbylabor-Handbuch [1] bietet einen ziemlich vollständigen elektrotechnischen Grundkurs. Alle Experimente aus den Büchern der beschriebenen Baukästen lassen sich mit gängigen und leicht erhältlichen Bauteilen auf dem vorgestellten Steckbrett-Silberling aufbauen. Die Originalteile sind im jeweiligen Heft hinten in

der Teileliste aufgelistet. Heute ist es besser, die LS-Variante der ICs zu wählen. Der „E-Magnet 6V“ ([36789](#)) und der „Kompass“ ([36776](#)), die für die Magnetismus-Experimente benötigt wurden, sind heute leicht durch Alternativen zu ersetzen. Ein Bündel Steckbrettkabel vervollständigt das erforderliche Material.

Spannungsmessungen können heute mit einem einfachen Multimeter durchgeführt werden. Wer ein Oszilloskop besitzt, kann sogar die vielen Grafiken im hobbylabor-Buch visualisieren. Beste Dienste leistet dabei natürlich der „Steckbrett-Silberling“ mit den Messausgängen auf der Rückseite.

Seit mir die neuen DIY-Module zur Verfügung stehen, habe ich viel damit herumgespielt. Die jüngsten Experimente betrafen Monoflop- und Flipflop-ICs sowohl in TTL-Logik als auch in CMOS. Die Bemerkung auf der letzten Seite des Begleitbuchs zum IC-digital Praktikum fasst meine vergangene Sommerzeit gut zusammen: „Allein die Erprobung der Kombination Flipflop-Monoflop lässt die ganzen großen Ferien über keine Langeweile aufkommen!“ Und tatsächlich erweist sich die Überschrift „fischertechnik – und kein Ende“ dieses allerletzten Absatzes als wahr: Im Jahr 2022 ist diese Materie immer noch faszinierend, trotz der vielen Möglichkeiten, die uns Mikroprozessoren und Steuerungen heute bieten.

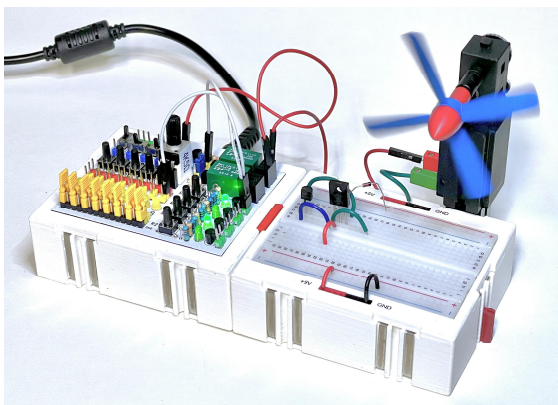


Abb. 14: Steuern von Lampen oder Motoren mit einem BD135 oder dem Relais auf dem Experimentierboard

Epilog

Eher exotisch sind die kleinen grünen ITT-Relais auf dem Peripherie-Silberling. Ich hatte einen ganzen Satz davon von der „Reststoffabteilung“ von AT&T/Philips in Hilversum, wo ich viele Jahre gearbeitet habe. Das sind Relais von Telefonzentralen. Es sind sehr leise und kompakte Relais, die maximal 1 A schalten können. Ich gebe zu, ich war von Nostalgie geleitet, als ich mich entschied, sie zu verwenden.

Kontaktiert mich gerne, wenn euch dieses Projekt anspricht oder wenn ihr Probleme habt, bestimmte Teile zu finden (E-Mail: arnoud@whizzbizz.com). Mein Vorrat an Platinen und Relais ist begrenzt, aber ich helfe gerne – auch beim 3D-Druck von „Silberling“-Gehäusen oder diversen anderen Teilen, die im Zusammenspiel mit fischertechnik zum Einsatz kommen [9].

Quellen

- [1] fischertechnik: [hobbylabor 1: Elektronik-Grundkasten](#). 1974.
- [2] fischertechnik: [Elektronik-Praktikum](#). 1977.
- [3] fischertechnik: [IC-Digital-Praktikum](#). 1977.
- [4] Arnoud van Delden: *Der Zauberling (Teil 4)*. [ft:pedia 1/2022](#), S. 71–79.
- [5] Arnoud van Delden: *Eine zukunftssichere Stromversorgung*. [ft:pedia 2/2022](#), S. 73–86.
- [6] ftCommunity Forum: [Silberlinge](#).
- [7] Dirk Fox: *Die Ewigkeitsmaschine*. [ft:pedia 1/2015](#), S. 41–43.
- [8] Dirk Fox, Thomas Püttmann: *Technikgeschichte mit fischertechnik*. dpunkt-Verlag, 2015 (Nachdruck 2021).
- [9] whizzbizz: [Design and printing of 3D parts](#).

Elektronik

Die Kunst der H-Brücke: ... and roll!

Thomas Magin

Wie schon im ersten Teil [1] angekündigt, soll der Aufbau einer Brückenschaltung so nah wie möglich an fischertechnik-Standards geschehen. Der Ausgangspunkt ist die Leistungsstufe.

Die Leistungsstufe werden wir ein wenig „pimpen“, indem wir die beiden NPN-Darlington-Transistoren durch je einen N-Kanal-MOSFET vom Typ IRFZ34N und einen P-Kanal-MOSFET vom Typ IRF9Z34N ersetzen. Man beachte den subtilen und keineswegs eingängigen Unterschied in der Typenbezeichnung.

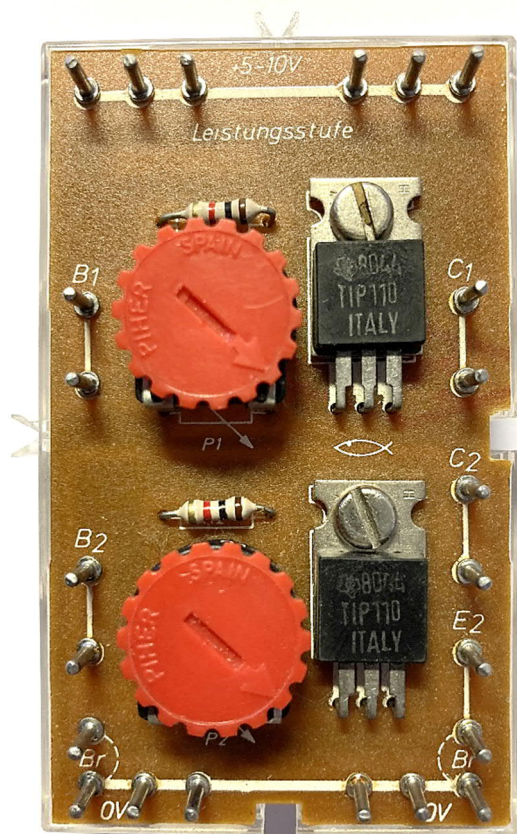


Abb. 1: fischertechnik-Leistungstufe mit NPN-Darlingtontransistoren

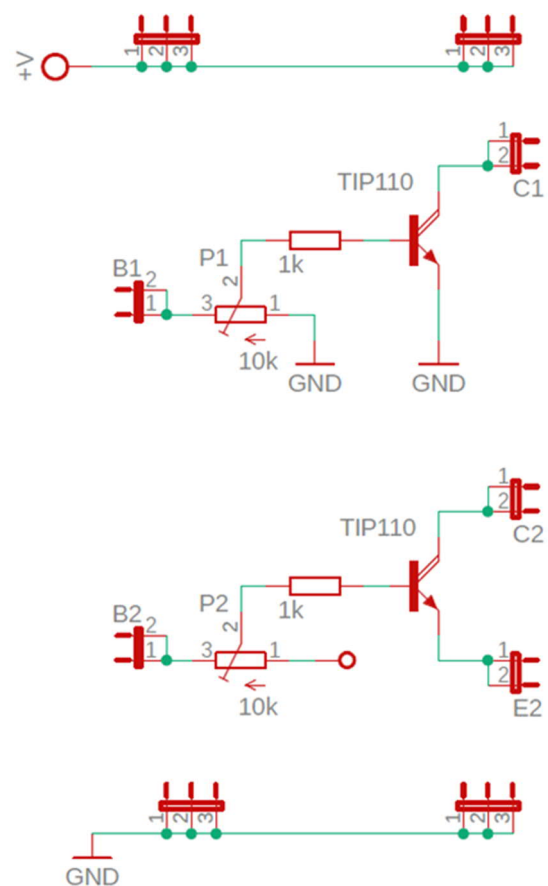


Abb. 2: Schaltplan der Leistungsstufe

The Heat is On

Zeit, das Werkzeug zu holen und den Lötcolben anzuheizen. Glücklicherweise hat uns fischertechnik das Leben hier einfach gemacht. Die Platinen sind nur einseitig beschichtet, haben insbesondere keine Durchkontaktierungen, trotzdem Lötstopplack. Da ist das Arbeiten ein Kinderspiel und absolut anfängergeeignet. Dazu kommt, dass beim unteren Transistor der

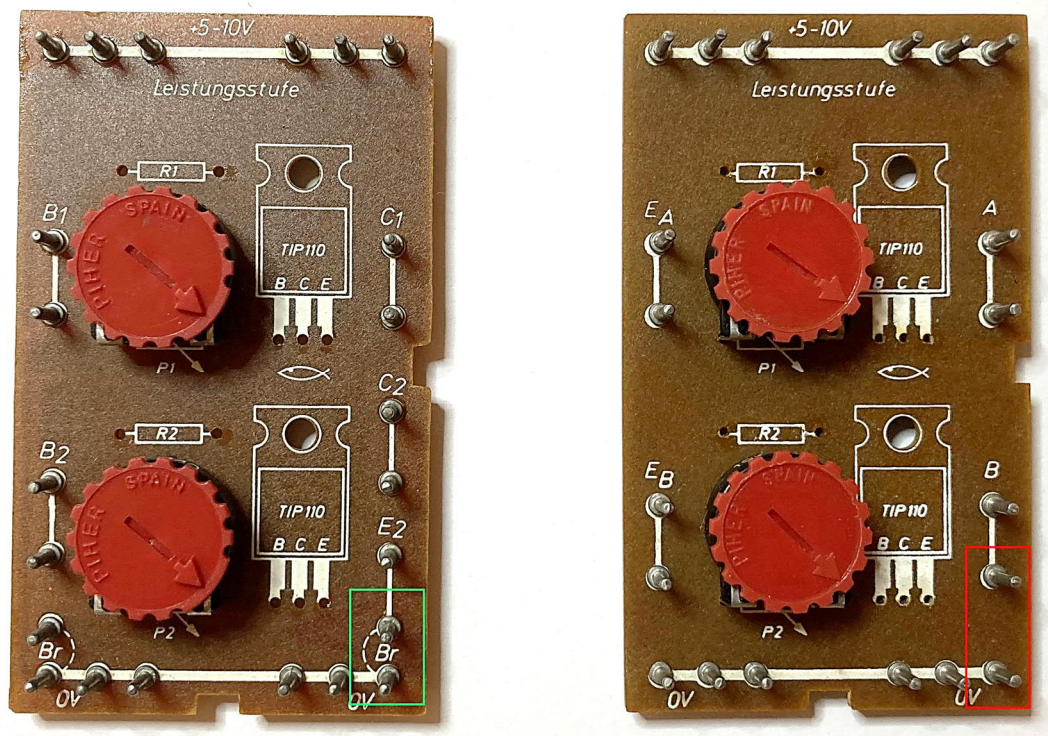


Abb. 3: Platinen mit flexibler und fester Verdrahtung des unteren Emitters

Emitter nicht direkt, sondern über eine Steckbrücke mit GND verbunden ist – sehr praktisch, denn beim P-Kanal-MOSFET (der High-Side) soll Source (die Analogie zum Emitter beim Bipolar-Transistor) ja an die positive Versorgungsspannung und nicht an GND. Was ist zu tun? Wir müssen

- Die beiden Transistoren und Widerstände entfernen,
- (die Platine modifizieren,)
- durch neue Typen ersetzen,
- zusätzlich zwei Dioden und zwei Kondensatoren einbauen
- und das Ganze in Betrieb nehmen

„Die Platine modifizieren“? Wie denn, wo denn, was denn? Ja, es gibt die Leistungsstufe (leider) auch in einer seltenen Version, bei der beide Emitter fest auf GND verdrahtet sind. Der Unterschied zeigt sich in Abb. 3.

Die zweite Variante scheint recht selten zu sein. Unter zehn Modulen in meinem Bestand zeigt gerade mal eine dieses Layout.

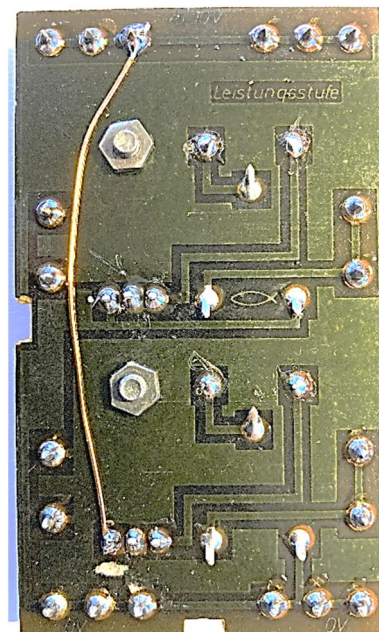


Abb. 4: Modifizierte Lötseite – Kugelfräser, Fädeldraht, fertig

Nichtsdestotrotz: Hier muss Hand angelegt werden!

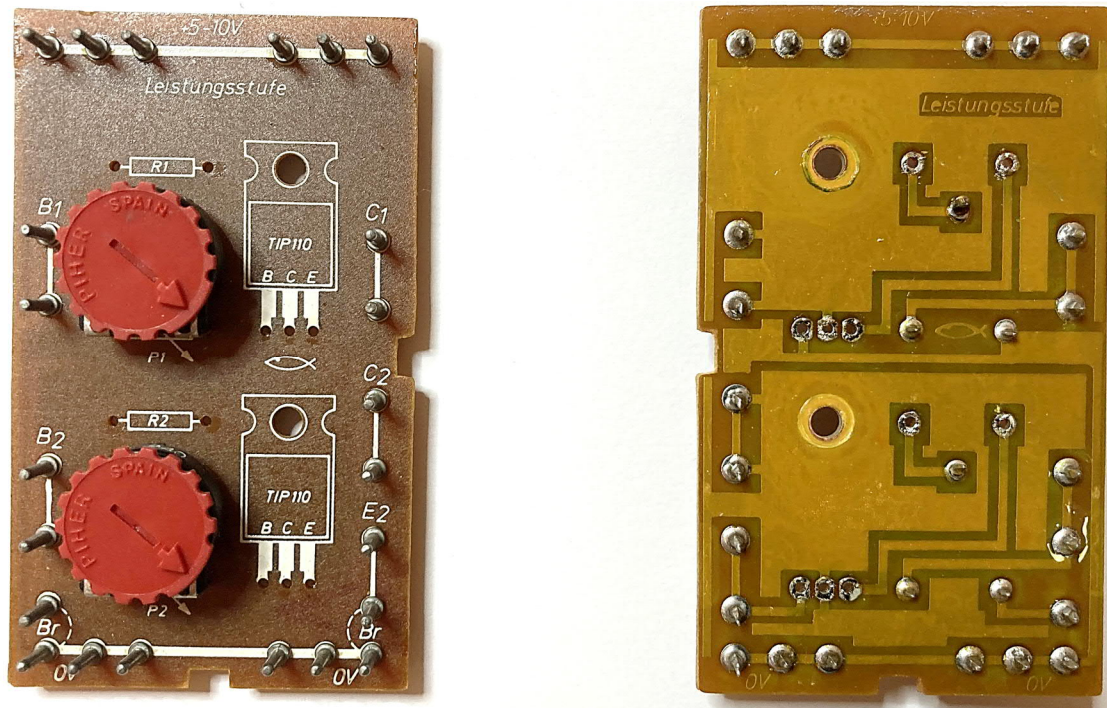


Abb. 5: Für die Neubestückung vorbereitete Platine

Die GND-Verbindung auf der Lötseite ist aufzutrennen und der Emitter-Anschluss mittels Fädeldraht auf VCC zu legen, also auf die „+5-10V“-Schiene des Moduls.

So oder so, jetzt kommen die Bauteile raus, und zwar die beiden Widerstände und beide Transistoren. Mit einer Lötabsaugpumpe, Entlötlitze oder rabiät mit dem Seitenschneider rücken wir unseren Veteranen zu Leibe und freuen uns über das Ergebnis in Abb. 5. Bei der Gelegenheit löten wir noch schnell die Anschlüsse der beiden Potis nach. Warum das? Sollten die Module Jahre und Jahrzehnte gestapelt gelagert worden sein, brechen die Lötstellen der höchsten Bauteile. Zwei solcher Platinen haben mich schlaflose Nächte gekostet, da sie mal funktionierten und mal nicht – bis ich herausfand, dass besagte Lötstellen gerissen waren. Ein paar Sekunden habe ich nachgelötet und alles war wieder in bester Ordnung.

Die beiden Potis bitte ganz nach **rechts** drehen. Damit liegt der Schleifer direkt und ohne Abschwächung an den Eingangspins – ganz wichtig!

Know Your Place

Die Bestückung ist schnell erledigt. Zur Erinnerung zeigt Abb. 6 noch einmal den Schaltplan, der auf zwei fischertechnik-Modulen umgesetzt wird (je Modul eine Halbbrücke).

Zuerst zu den Transistoren. Der N-Kanal-MOSFET IRFZ34 wird oben auf dem Modul bestückt, da bei diesem der Source-Anschluss immer fest mit GND verbunden ist. Im Schaltbild findet sich der Transistor unten, wo er als echter „Low-Side“ auch hingehört. Der P-Kanal-Typ IRF9Z34 kommt auf der Platine dagegen auf den unteren Bestückungsplatz. In diesem Arbeitsgang werden dann gleich noch die 47-k Ω -Gate-Widerstände mit eingelötet.

Nun müssen wir die Platine umdrehen und auf der Lötseite die Kondensatoren und Dioden bestücken. Bei den Dioden bitte auf korrekte Polung achten! Bei den Kondensatoren ist man in der Typenwahl recht frei – irgendetwas Keramisches funktioniert. Müssen das wie in Abb. 7 3-kV-Typen

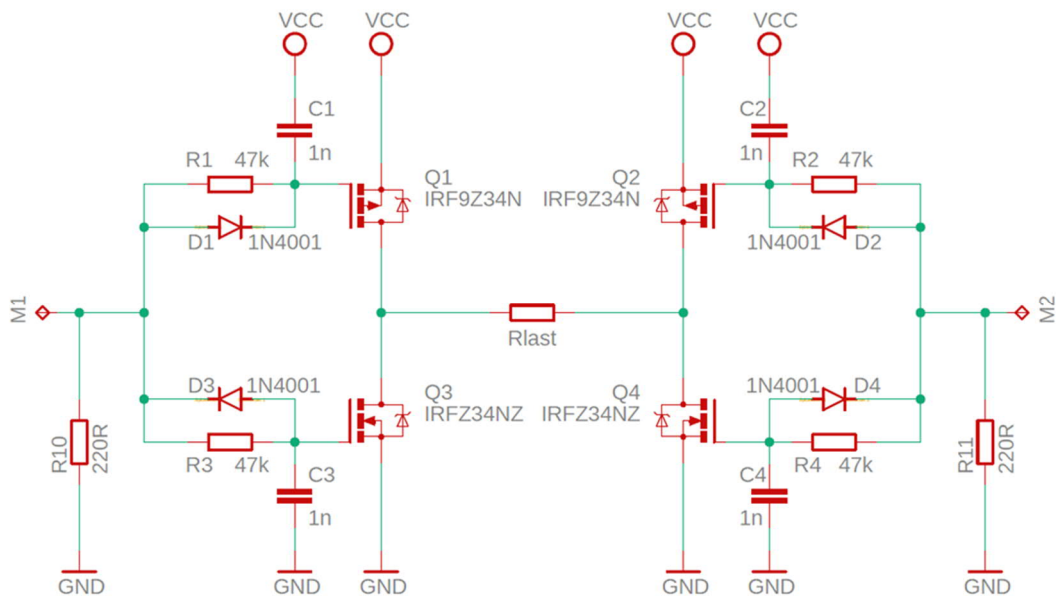


Abb. 6: Schaltplan der H-Brücke (ohne Levelshifter)

sein? Natürlich nicht. Aber zum einen passen die wegen des Rastermaßes so schön, zum anderen hatte ich sie in der Bastelkiste liegen. Und so sind wir dann auch schon beim finalen Ergebnis.

Another Brick for Your Wall

Jetzt stellt sich die Frage, wie man die modifizierten Leistungsstufen schick in fischertechnik integrieren kann. Mein Plan war eine Art Einschubsystem, in welches ich verschiedenste Module integrieren kann. Also begann ich damit, eine

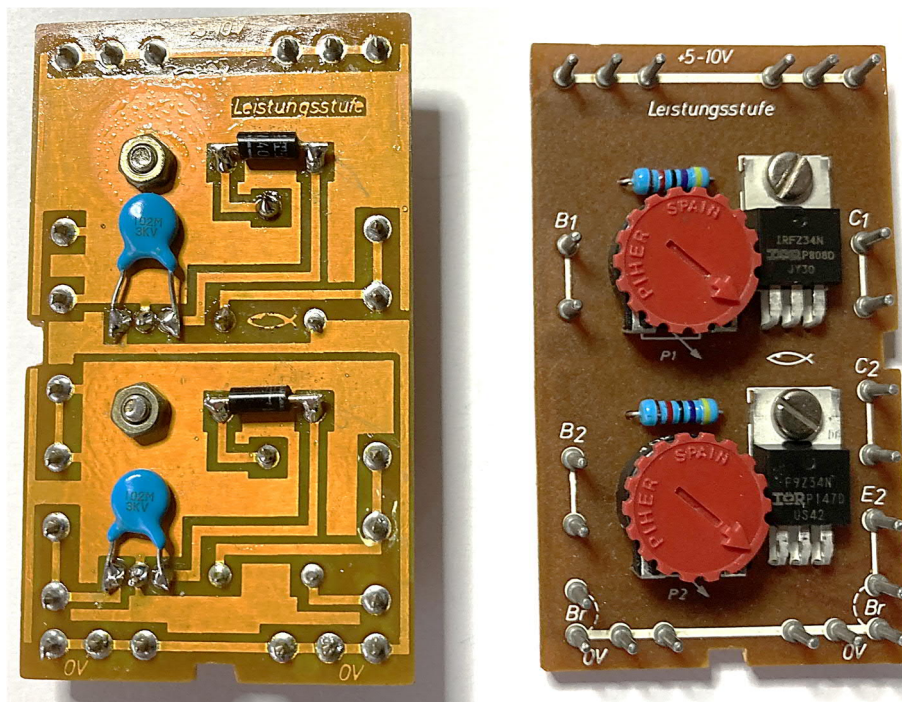


Abb. 7: Die fertig bestückte Platine

Leistungsstufe (später noch eine Relaisstufe) auf einer Grundplatte [31001](#) aufzubauen, wo auch die Levelshifter ihren Platz fanden. Um letztere zusammenzustecken, wählte ich Stecksocket ([38230](#) + [38229](#) + [37918](#)) – ein recht wackeliger Aufbau, wie ich zugeben muss. Da sind Versuche mit der

Transistoranschlussplatte ([152218](#)) vielversprechender. Die Verdrahtung erfolgt mit passenden Leitungen, soweit möglich aus fischertechnik-Beständen. Hin und wieder muss man dann aber auch selbst zu Litze, Lötflanke und LötKolben greifen.

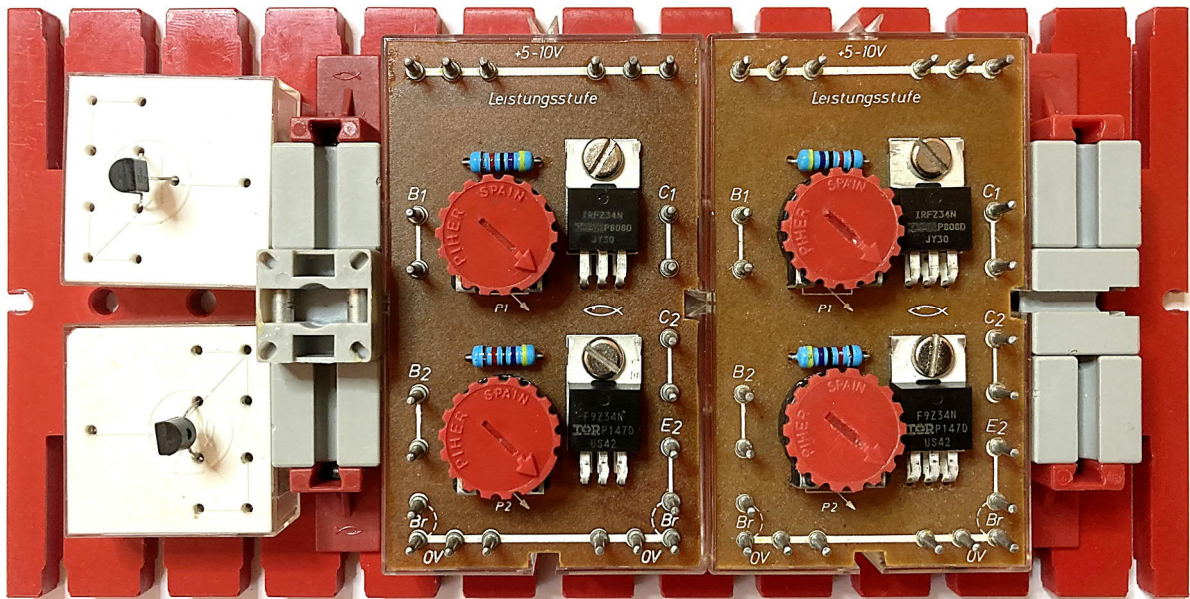


Abb. 8: Anordnung der Bauelemente

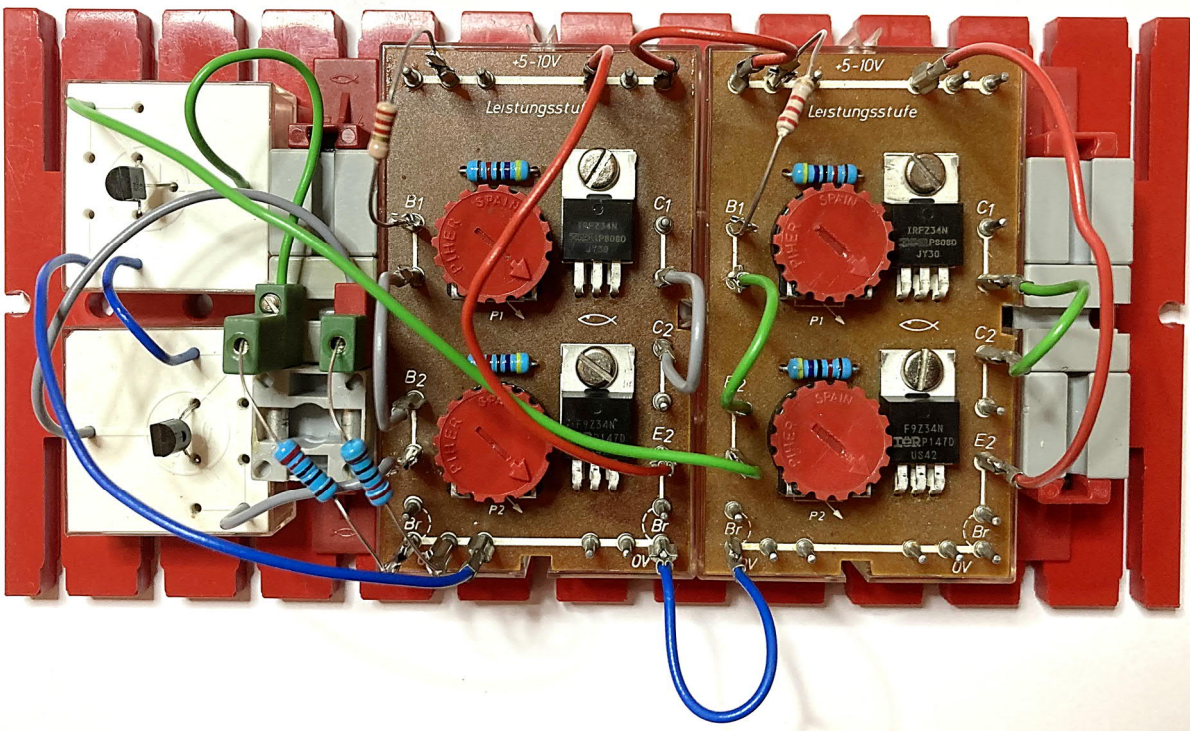


Abb. 9: Verdrahtung

Something Got Me Started

Jetzt folgt der spannendste Teil, die Inbetriebnahme! Kleiner Tipp: Niemals abends kurz vorm Schlafengehen – nie! Denn wenn etwas schiefgeht, und potenzielle Müdigkeit tut ihr Übriges, bringt es einen um den Rest der Nacht. Ich weiß, hard to resist...

Also ist eine 9 V-Stromversorgung anzuschließen. Wie im vorherigen Teil angedeutet, am besten eine geregelte und abgesicherte. Und wenigstens 1 A sollte die schon liefern können. Als Last ist ein Leistungswiderstand von $20\ \Omega$ erstmal ein guter Anfang. Als Eingangssignal dient ein Funktionsgenerator, der an einen der beiden Eingänge gehängt wird: Ein positiver Rechteckimpuls variabler Breite, 200 Hz, 9 V. Dank Levelshifter sind wir bei der Spannung aber flexibel.

Dann wird ein Oszilloskop an den korrespondierenden Ausgang angeschlossen. Je nach persönlicher Neugierde kann man weitere Kanäle an die beiden Gates der Transistoren anschließen, um den gesamten Signalverlauf im Blick zu behalten. Und wenn man anständig gearbeitet hat, stellt sich in etwa so ein Bild auf dem Oszilloskop ein wie in Abb. 11.

Poor Man's Blues

Und wenn ich keinen Funktionsgenerator habe? Und schon gar kein Oszilloskop? Dann nehmen wir einen TX(T), ein wenig ROBO Pro-Code [7] und ein Multimeter. Letzteres ist heutzutage ja gar nicht mehr ohne True-RMS [2, 3] zu bekommen, was jetzt eine große Hilfe ist.

Gnädiger Weise lässt sich der Effektivwert einer gepulsten Spannung ganz ohne Studium und Bronstein [4] errechnen. Wir

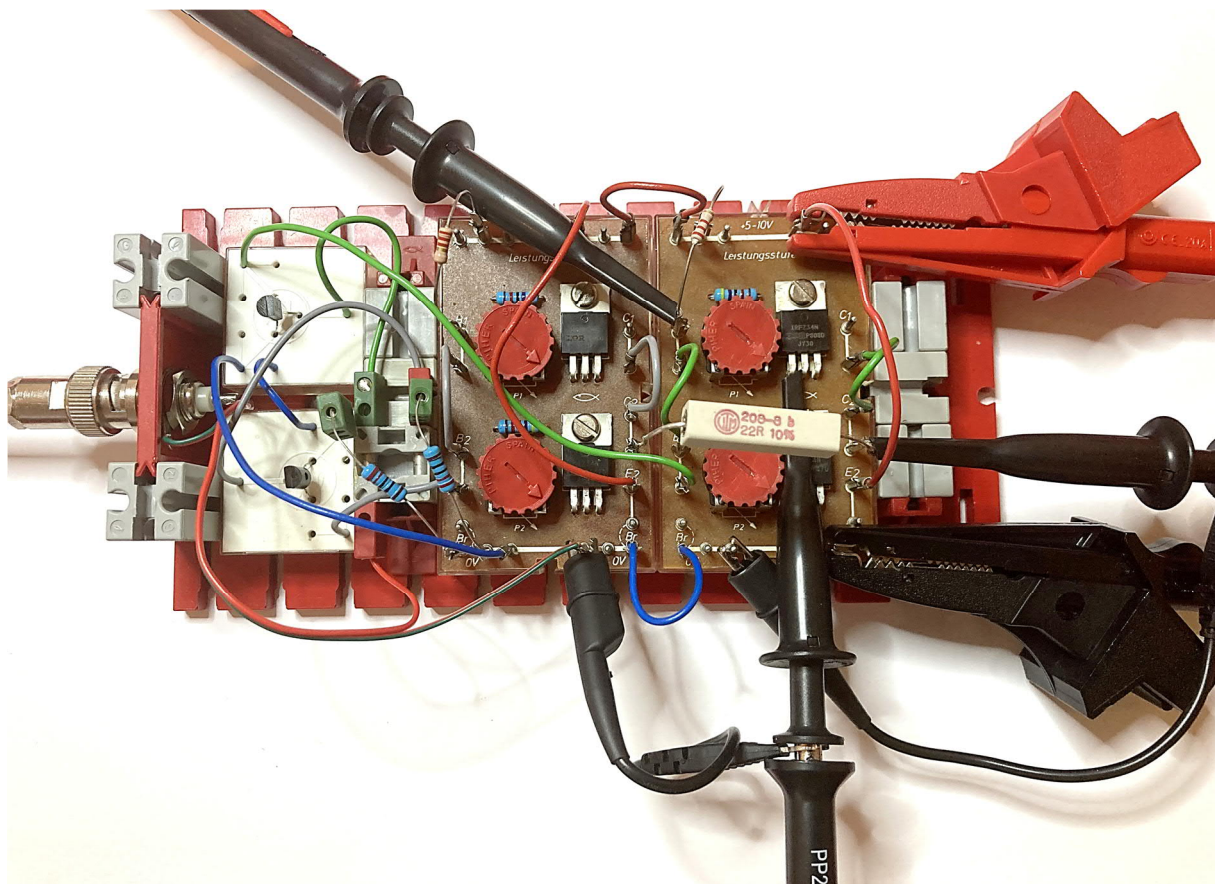


Abb. 10: Inbetriebnahme mit Lastwiderstand

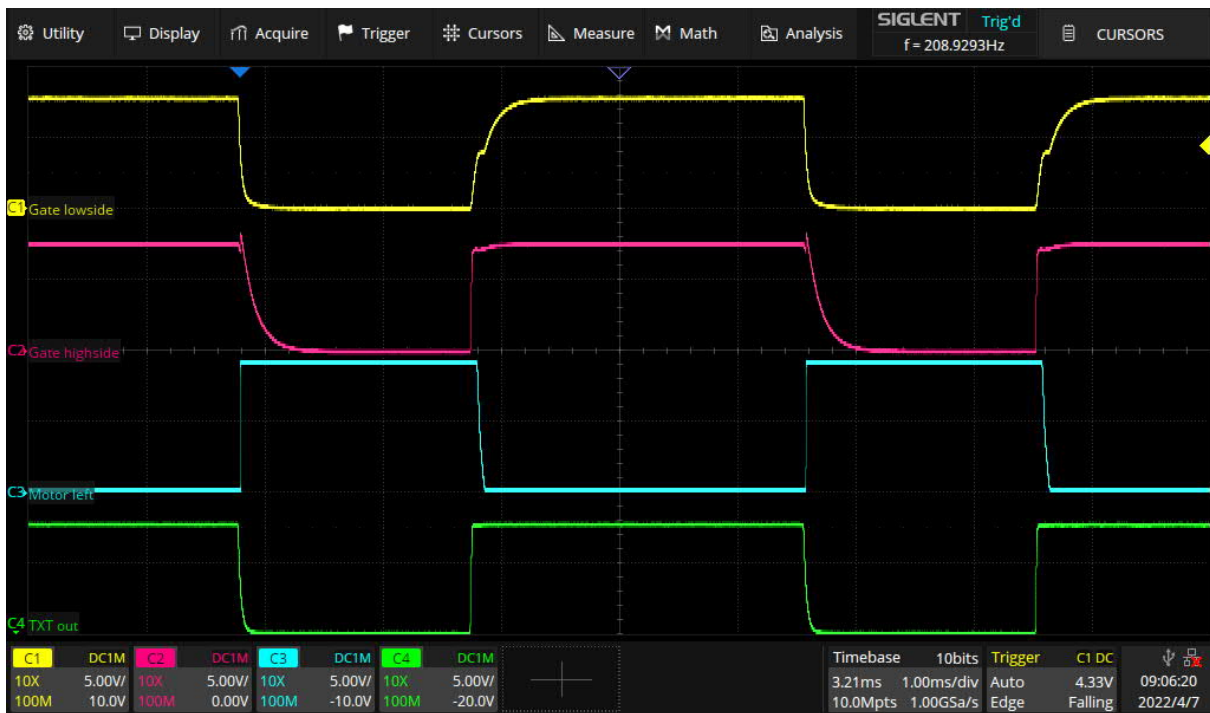


Abb. 11: Heureka

erinnern uns (nicht): Der Effektivwert einer Spannung entspricht jener konstanten Gleichspannung, die die gleiche Energie wie die Wechselspannung liefert. Was in unserem Falle der Duty cycle [5, 6] multipliziert mit der Spitzenspannung ist. Der Duty cycle in Prozent ergibt sich aus dem Quotienten T_{ON} / T und wird durch den TX(T) vorgegeben. Wenn man für den Motor die 8-Schritt-Auflösung wählt, liegen die Duty cycle bei 13 %, 17 %, 21 %, 26 %, 33 %, 42 %, 61 % und 100 %. Mal Eingangsspannung – that's it for U_{eff} . Das Ganze ist in ROBO Pro für 9 V hinterlegt.

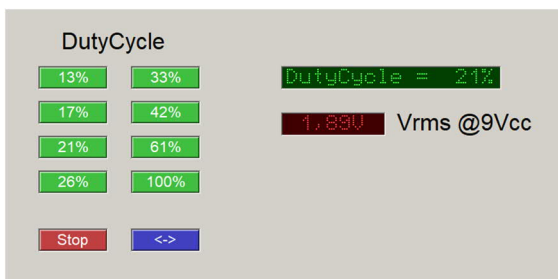


Abb. 12: ROBO Pro UI

Mit den Prozent-Tasten stellt man den gewünschten Duty cycle ein. Der angezeigte Spannungswert stellt das Soll der

TX(T)-Ausgabe dar. Und das vergleichen wir jetzt einfach mit der Anzeige unseres Multimeters. Bitte die RMS-Einstellung nicht vergessen.

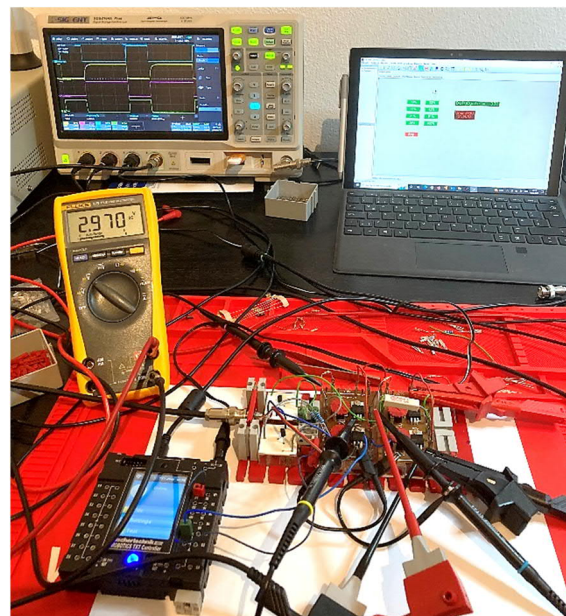


Abb. 13: Messaufbau

In der Realität passt das recht gut. Der „<->“-Taster wechselt die Laufrichtung, was in unserer Anwendung bedeutet, dass O1 oder O2 aktiv sind.

Wrap it up

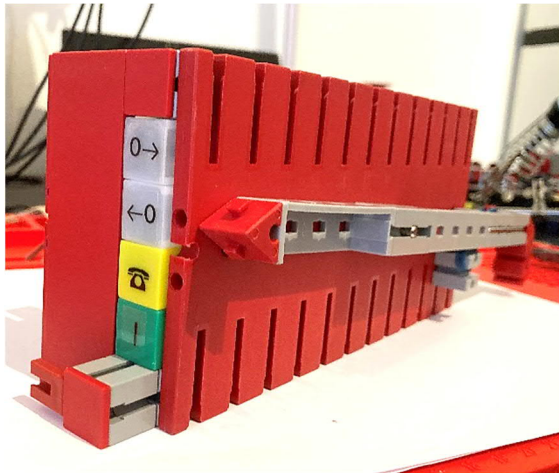


Abb. 14: Vorderseite mit Status-LEDs

Jetzt wollen wir der H-Brücke noch eine schöne Verpackung geben. Da sind der Fantasie natürlich keine Grenzen gesetzt. In meinem Fall habe ich dem Modul mit einer zweiten Grundplatte einen Deckel aufgesetzt (und den Tag verflucht, an dem sich fischertechnik mal wieder nicht ans Rastermaß gehalten hat). Drei Grundplatten übereinander sind nicht so hoch wie ein 15er-Baustein. Dies und seine Konsequenzen bei der Integration in ein Einschubsystem (dafür auch der angeflanschte Arm an der Seite) sind aber eine andere Geschichte.

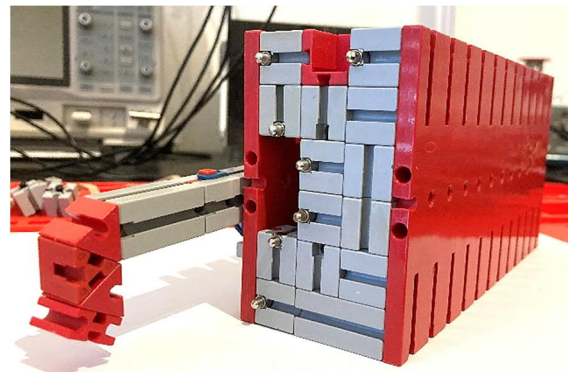


Abb. 15: Rückseite mit Federkontakten

Quellen

- [1] Thomas Magin: *Die Kunst der H-Brücke: Let's Rock*. [ft:pedia 2/2022](#), S. 104–118.
- [2] Wikipedia: [Effektivwert](#).
- [3] Fluke Corporation: [What is true-RMAS?](#)
- [4] Wikipedia: [Taschenbuch der Mathematik](#).
- [5] Wikipedia: [Duty cycle](#).
- [6] Wikipedia: [Tastgrad](#).
- [7] Thomas Magin: *PulseGenerator.rpp*. ROBO Pro-Programm im [Downloadbereich zu dieser Ausgabe](#) der ft:pedia.

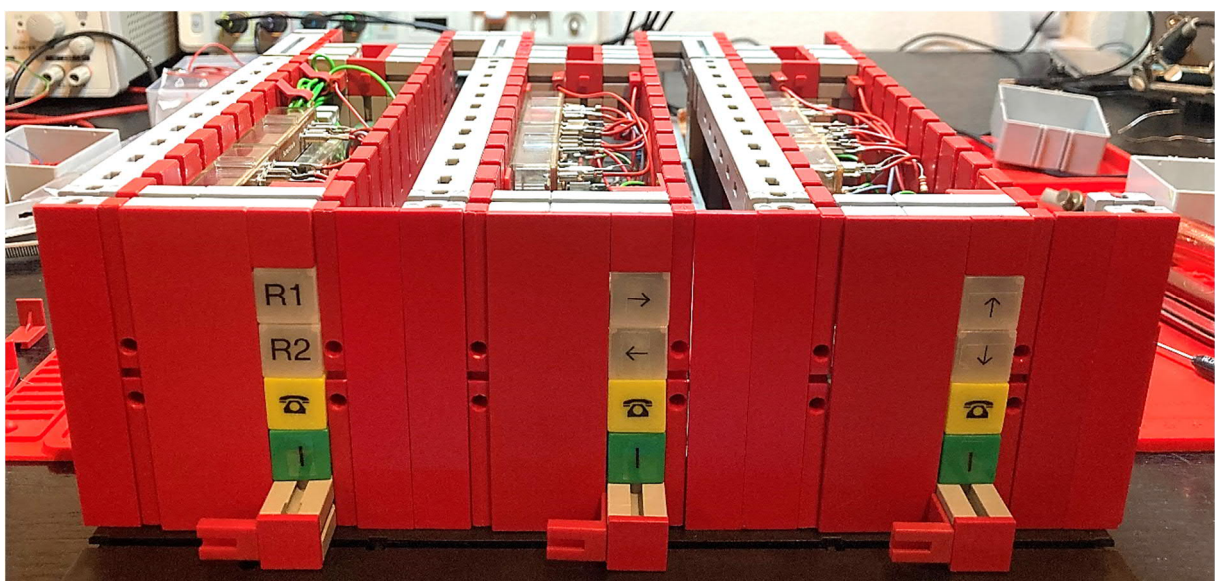


Abb. 16: Einschubsystem

Elektronik

Silberlinge: Original oder Nachbau (Teil 8)

Peter Krijnen

1971 stellten die Fischerwerke die erste Version des Schulkastens u-t4 [30609](#) (Abb. 230) vor. Diese war speziell zum Steuern und Regeln gedacht. Die notwendige Elektronik wurde in einem „Batteriehalter“ untergebracht. Ab 1974 war der Kasten jedoch mit vier Silberlingen (Abb. 231) versehen. Die Elektronik, die im ersten Kasten in einem Gehäuse untergebracht war, war fortan auf drei Silberlinge verteilt: Verstärkerbaustein [36733](#), Transistor-Potentiometer-Baustein [36735](#) und ein Relaisbaustein [36734](#). Ein Gleichrichterbaustein [36393](#) war ebenfalls dabei.

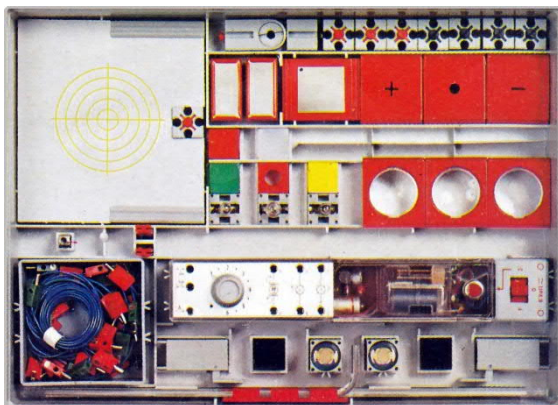


Abb. 230: Erste Version des Schulkastens u-t4 aus 1971

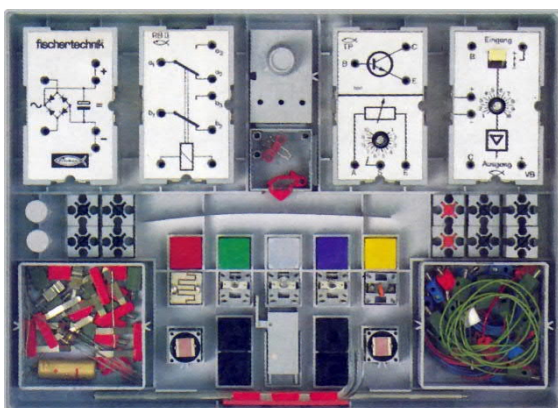


Abb. 231: Zweite Version des Schulkastens u-t4 aus 1974

Verstärkerbaustein VB [36733](#)

Wenn wir auf die Frontplatte des Moduls schauen, sehen wir insgesamt 6 Anschlussbuchsen, einen Schalter, die Skala eines Potentiometers und das Symbol eines Verstärkers. Oben steht: „Eingang“ und unten: „Ausgang“. Links neben dem Schalter steht „B“ und rechts „+/-“. Links neben dem Verstärkersymbol steht „C“ und rechts „+“.

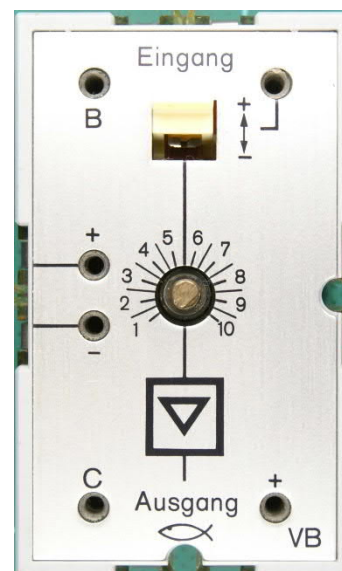


Abb. 232: Verstärkerbaustein VB [36733](#)

„B“ (Basis) und „C“ (Kollektor) zeigen an, dass ein Transistor eingebaut ist. Ihr fragt euch vielleicht, wo sich der Anschluss „E“ (Emitter) befindet? Und auf der linken Seite

des Moduls gibt es auch noch einen „+“ und einen „-“-Anschluss.

Wenn wir uns nun das Schaltbild (Abb. 236) ansehen, sehen wir, dass das „B“ und das „C“ tatsächlich zu einem Transistor gehören. Wir sehen jetzt auch, dass das „E“ des Transistors mit „-“ des Moduls verbunden ist. Und dass der Schalter, wie der Polwendeschalter [31331](#), zwei Wechslerkontakte hat.

Einer der Kontakte wird über die Anschlussbuchse „+/-“ herausgeführt. Der andere Kontakt ist über den 330- Ω -Widerstand R1 mit dem Schleifer des Potentiometers verbunden. Allerdings ist der Effekt umgekehrt, „-/+“ also. Dies wird auf Seite 8 der Anleitung [1], die mit u-t4 geliefert wird, besser beschrieben.

Einer der Anschlüsse des 25K-Potentiometers ist mit der Anschlussbuchse „B“ und zusätzlich über den 1K-Widerstand R2 mit der Basis des Transistors verbunden. Da der Emitter an „-“ angeschlossen ist, muss die Last (Lampe, Motor, Relais) zwischen Kollektor und „+“ angeschlossen werden.

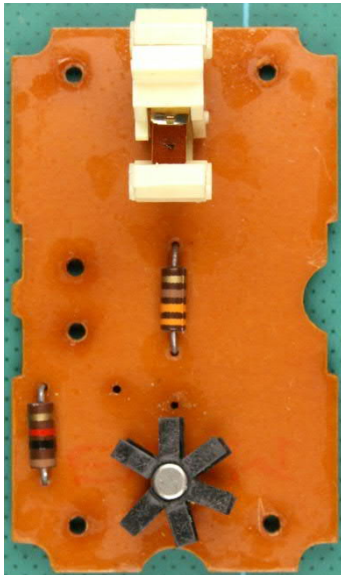


Abb. 233: Platine des VB

Es muss sichergestellt sein, dass sich der Motor frei drehen kann, da sonst der Strom zu stark ansteigen kann. Dadurch wird der Transistor irreparabel beschädigt. Der

BC108 verträgt nur 200 mA. Da ich selbst mehrfach nicht aufgepasst hatte, musste ich mehrfach den Transistor tauschen. Da die Verfügbarkeit des BC108 nicht mehr so gut ist, habe ich ihn durch einen BC517 ersetzt. Dies ist ein 400 mA-Darlington in einem Kunststoffgehäuse. Der BC108 hat ein rundes Metallgehäuse und ist mit einem Kühlstern ausgestattet.

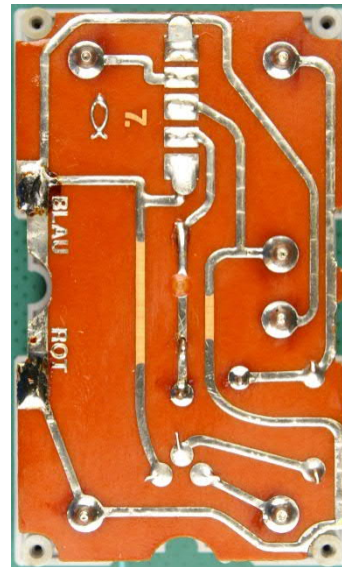


Abb. 234: Leiterbahnseite des VB

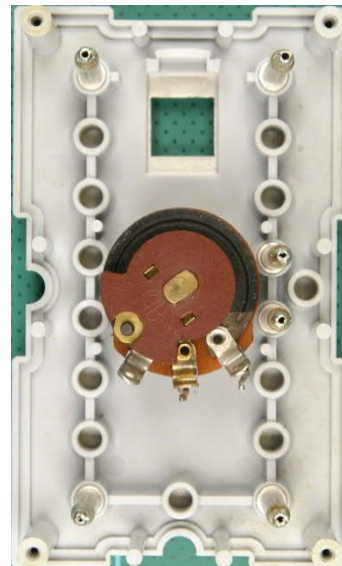


Abb. 235: Unterseite des Deckels

Ich möchte kurz auf den Widerstand R1 zurückkommen. Mir ist aufgefallen, dass dieser Widerstand ursprünglich nicht vorgesehen war. Wenn wir uns Abb. 234 ansehen, sehen wir, dass eine Leiterbahn in

der Mitte der Platine durchbohrt wurde. Wir sehen auch keine Lötinseln für den Widerstand. Die Drähte des Widerstands werden direkt auf die Leiterbahn gelötet.

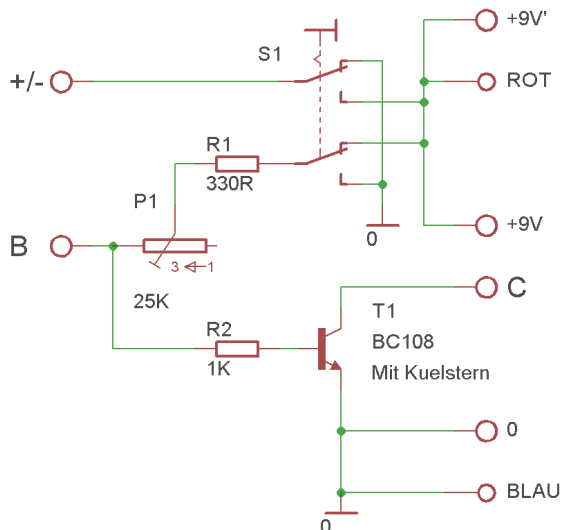


Abb. 236: Schaltplan des VB

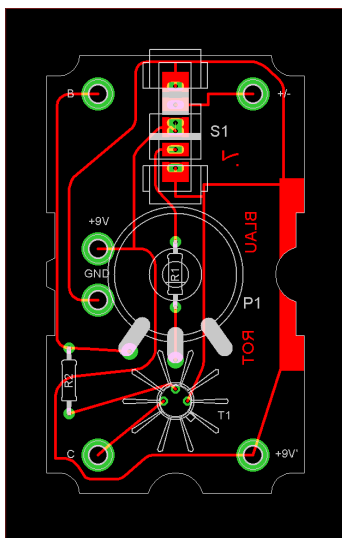


Abb. 237: Layout des VB

Transistor-Potentiometer-Baustein TP [36735](#)

Der TP ist eigentlich ein VB ohne Schalter.

Auf dem Schaltbild (Abb. 242) sehen wir, dass der Transistor ebenfalls vom Typ BC108 und ebenfalls mit einem Kühlstern ausgestattet ist. Das 25K-Potentiometer ist auch das gleiche. Anders ist, dass nun alle

Anschlüsse, sowohl die des Transistors als auch die des Potentiometers, mit einer Anschlussbuchse nach außen geführt sind. Der BC108 könnte durch den BC517 ersetzt werden.

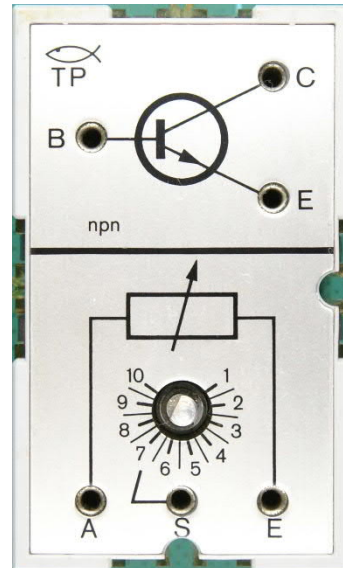


Abb. 238: Transistor-Potentiometer TP [36735](#)

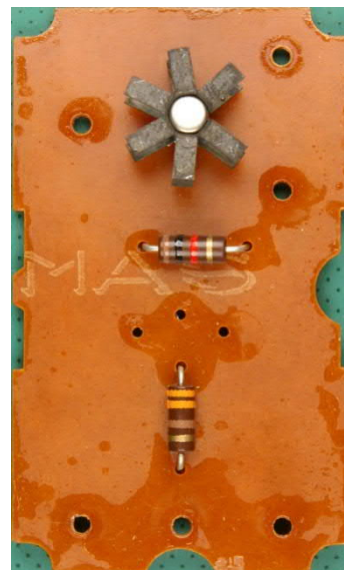


Abb. 239: Platine des TP

Seite 15 von [1] zeigt, wie es möglich ist, die Transistoren des VB und des TP in Reihe zu schalten. Dies ergibt einen 400 mA-Darlington.

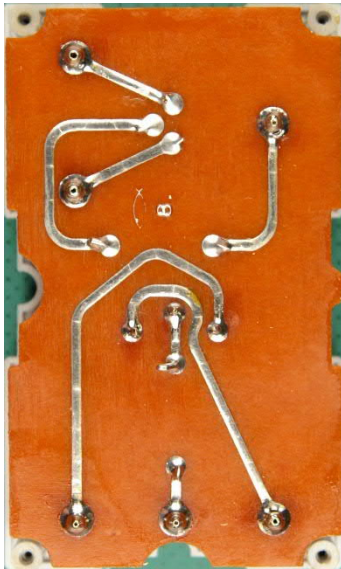


Abb. 240: Leiterbahnseite des TP



Abb. 241: Unterseite des Deckels

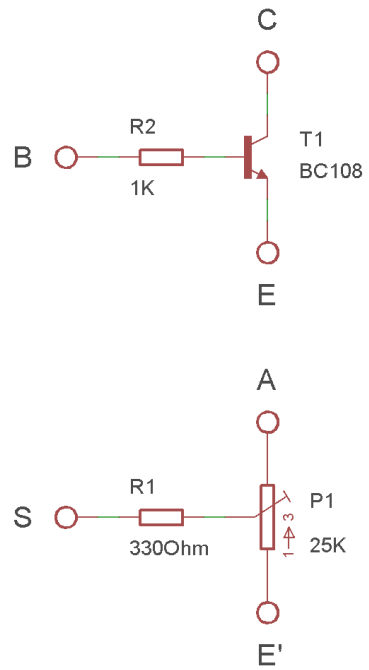


Abb. 242: Schaltplan des TP

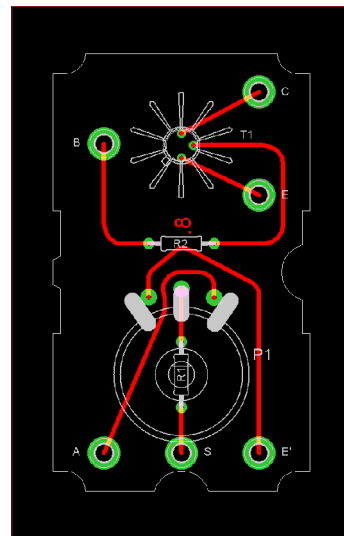


Abb. 243: Layout des TP

Relaisbaustein RB [36734](#)

Äußerlich ähnelt der Relaisbaustein [36734](#) dem Relaisbaustein RBII [37683](#). Es gibt jedoch Unterschiede. Da nun vier Dioden (Abb. 245) verbaut sind, ist auch das Layout (Abb. 249) der Platine anders. Die Dioden sind als Gleichrichter geschaltet (Abb. 248). Das bedeutet, dass dieses Relais mit Wechselstrom geschaltet werden kann. Der Nutzen davon entgeht mir jedoch, zumal wir innerhalb des fischertechnik-Programms ausschließlich mit Gleichstrom/Gleichspannung arbeiten.

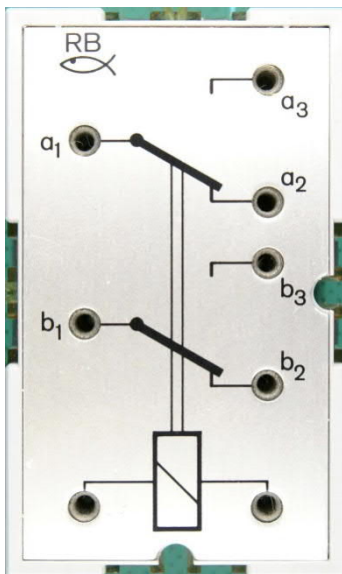


Abb. 244: RB – Relaisbaustein RB 36734

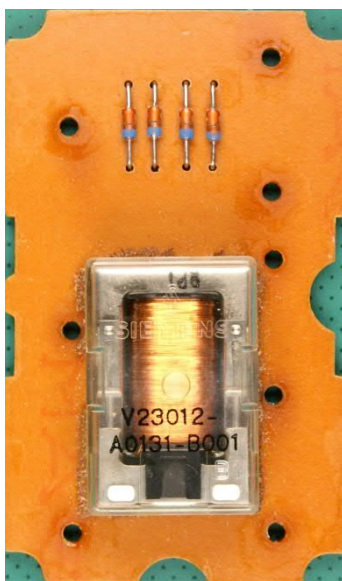


Abb. 245: Platine des RB

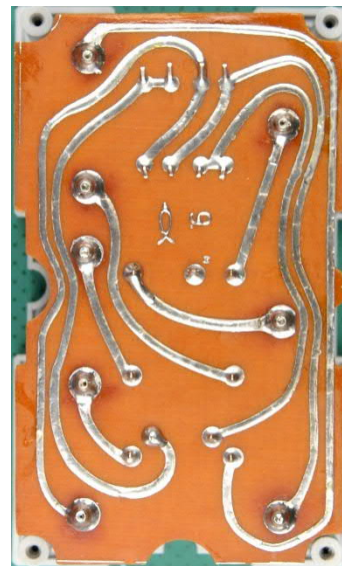


Abb. 246: Leiterbahnseite des RB



Abb. 247: Unterseite des Deckels

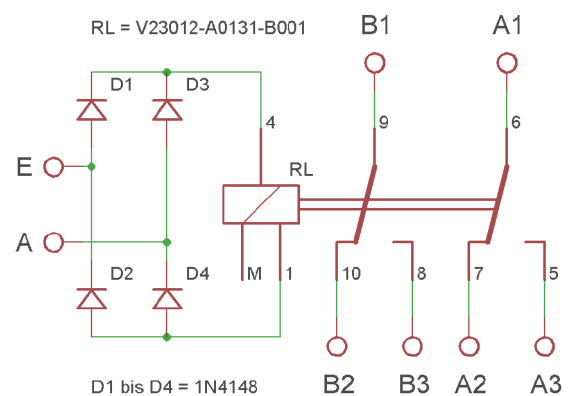


Abb. 248: Schaltplan des RB

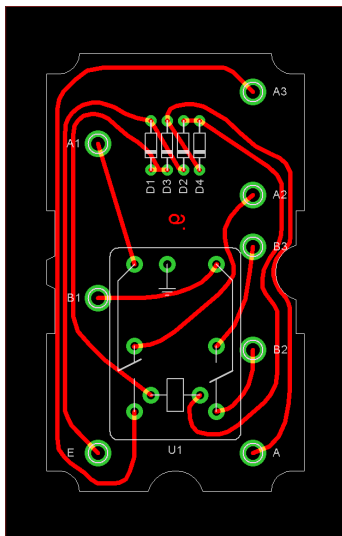


Abb. 249: Layout des RB

Nachbau

Wie ich in der Erklärung des Dynamischen-UND erwähnt habe [2], habe ich eine universelle Platine für die u-t4- und Hobby-labor-Module entworfen. Da diese Module nur aus wenigen Komponenten bestehen, wäre es mir (finanziell) unmöglich gewesen, für jedes Modul eine separate Platine anfertigen zu lassen. Mehr dazu in einem zukünftigen Beitrag.

Aufgrund des universellen Charakters der Platine habe ich den Emitter des VB über eine eigene Anschlussbuchse nach außen geführt (Abb. 250, 253). Auch habe ich die Anschlussbuchsen an eine für mich logischere Stelle verschoben.

Vergleicht man die Abb. 251 und 260 bzw. 254 und 263, sieht man, dass der Schalter der einzige Unterschied ist. Da das 9V-Batteriegehäuse etwas zu klein ist, musste ich für diese Versionen zwei Platinen anfertigen. Diese findet man auf den Abb. 257 und 266.

Einen Nachbau für den Relaisbaustein habe ich nicht geplant. Ich denke, die Version mit dem eingebauten Verstärker ist besser zu verwenden.

In diesem Beitrag gibt es also nicht viel Text, aber auf 38 Bildern trotzdem einiges zu sehen. Und ein Bild sagt schließlich mehr als 1000 Worte.

Im nächsten Beitrag wird auf das Hobby-labor eingegangen, ergänzt um den Mikrofon-Lautsprecher-Baustein.

Quellen

- [1] Fischer-Werke: *fischertechnik u-t4 Beschreibung und Anwendung der Bauelemente des Elektronikbaukastens*. Auf docs.fischertechnikclub.nl.
- [2] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 7)*. ft:pedia 2/2022, S. 87–100.



Abb. 250: Nachbau 2: Frontplatte des VB für das 45×75-Gehäuse

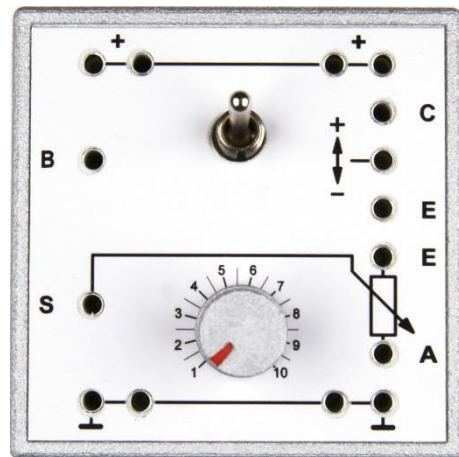


Abb. 253: Nachbau 3: Frontplatte des VB für das 60×60-Gehäuse



Abb. 251: Nachbau 2: Platine des VB für das 45×75-Gehäuse

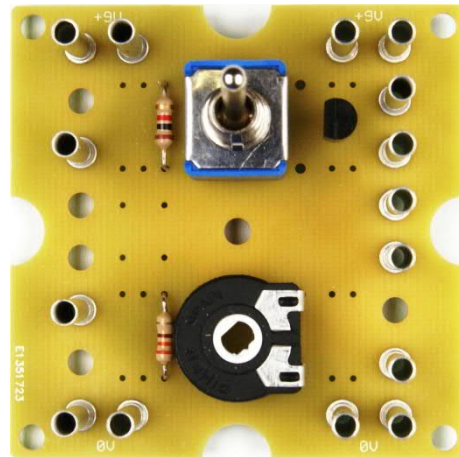


Abb. 254: Nachbau 3: Platine des VB für das 60×60-Gehäuse

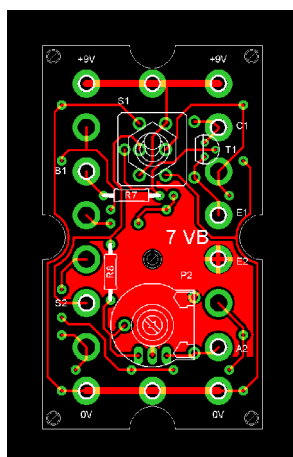


Abb. 252: Nachbau 2: Layout des VB für mein 45×75-Gehäuse

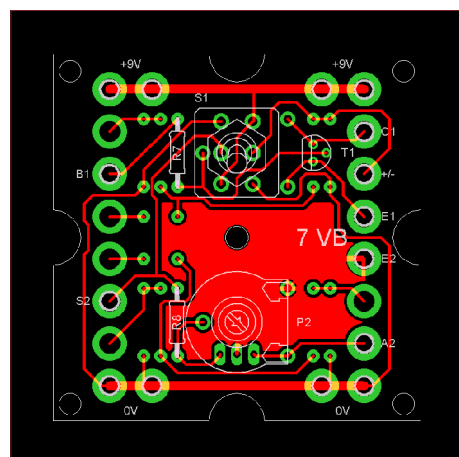


Abb. 255: Nachbau 3: Layout des VB für die 60er-Kassette ([32076](#))



Abb. 256: Nachbau 4: Frontplatte des VB für das Batteriegehäuse [32263](#)

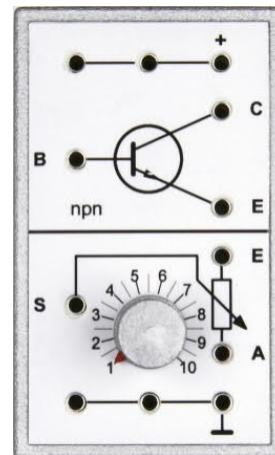


Abb. 259: Nachbau 2: Frontplatte des TP für das 45×75-Gehäuse

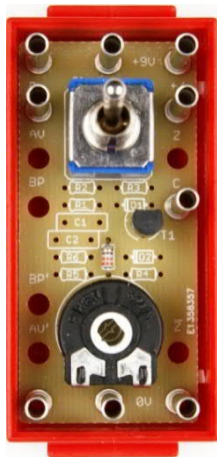


Abb. 257: Nachbau 4: Platine des VB für das 30×60-Gehäuse

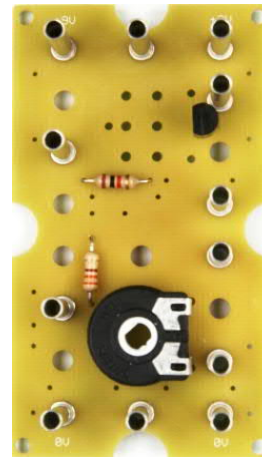


Abb. 260: Nachbau 2: Platine des TP für das 45×75-Gehäuse

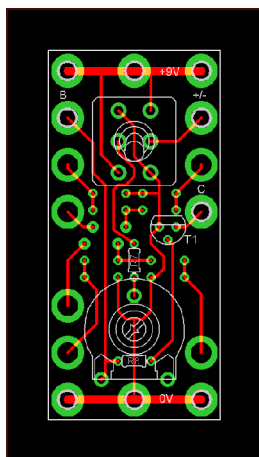


Abb. 258: Nachbau 4: Layout des VB für in Batteriegehäuse [32263](#)

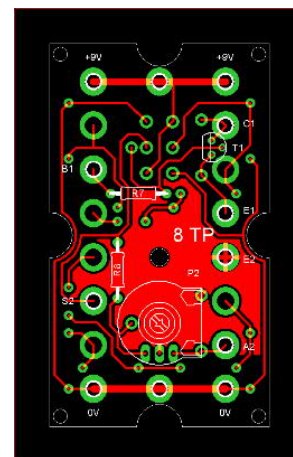


Abb. 261: Nachbau 2: Layout des TP für mein 45×75-Gehäuse

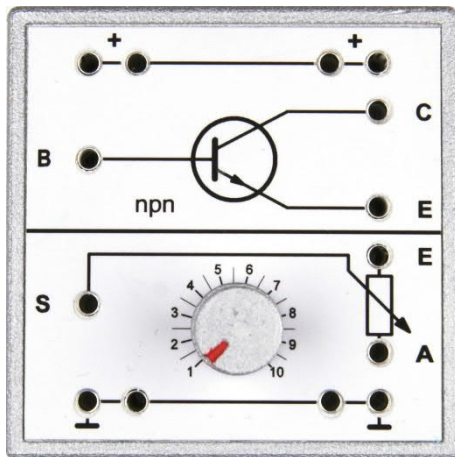


Abb. 262: Nachbau 3: Frontplatte des TP für das 60×60-Gehäuse

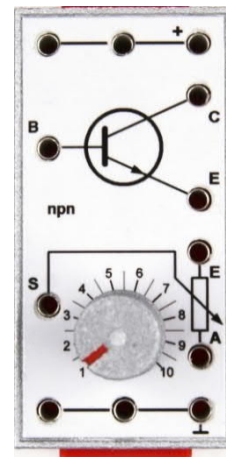


Abb. 265: Nachbau 4: Frontplatte des TP für das Batteriegehäuse [32263](#)

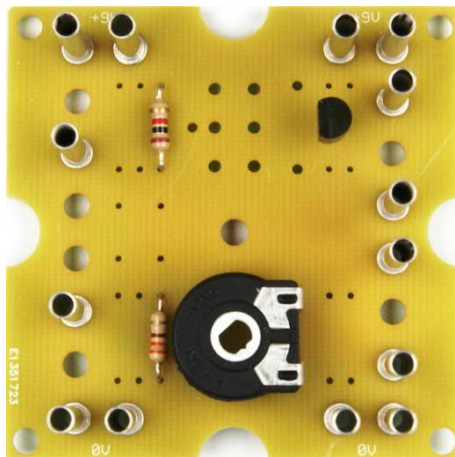


Abb. 263: Nachbau 3: Platine des TP für das 60×60-Gehäuse



Abb. 266: Nachbau 4: Platine des TP für das 30×60-Gehäuse

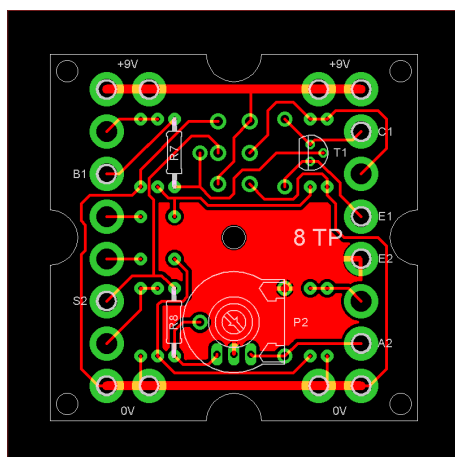


Abb. 264: Nachbau 3: Layout des TP für die 60er-Kassette ([32076](#))

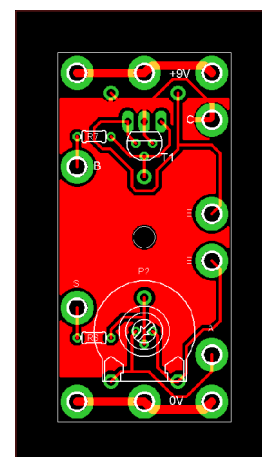


Abb. 267: Nachbau 4: Layout des TP für in Batteriegehäuse [32263](#)

Elektronik

Elektronik-Module (Teil 8): Silberlinge = Elektronikmodule?!

Hans-Christian Funke

Aufgrund vielfältiger Nachfragen, wie die Elektronikmodule eingesetzt oder diese ersatzweise für die Silberlinge verwendet oder wie die Elektronikmodule in einem Modell aus den hobby-Büchern integriert werden können, habe ich mich entschlossen, hier einmal einen allgemeinen Vergleich zu geben und zu zeigen, wie die einzelnen Silberlinge durch Elektronikmodule ersetzt werden können.

Einleitung

Es gibt zwei große Hürden beim Nachbau von Modellen mit Silberlingen aus Bauanleitungen, aus den hobby-Büchern oder anderen Quellen: Entweder fehlt einem ein Elektronikbaustein und dieser ist auch nicht bei eBay oder einem Händler zu bekommen oder man fällt schlichtweg in Ohnmacht, wenn man die Preise sieht.

Fischertechnik Elektronik MonoFlop Baustein h4MF	
Artikelzustand:	Gebraucht
Restzeit:	2T10Std
Startpreis:	EUR 69,00
	[0 Gebote]

Abb. 1: Mindestgebot für einen Mono-Flop-Silberling (h4MF) bei eBay.

Hier bieten die Elektronikmodule einen idealen Ersatz für die Silberlinge. Sie sind verfügbar [1] und bieten zu dem auch preislich eine gute Alternative. Ein Silberling h4MF enthält ein Mono-Flop – ein Elektronikmodul 70005 enthält *zwei* Mono-Flops mit dem gleichen Funktionsumfang. Bei direktem Vergleich muss man somit den Preis bei einem Elektronikmodul halbieren und man zahlt rund 20 € für ein Mono-Flop. Ähnlich verhält es sich mit den meisten

anderen Elektronikmodulen, die überwiegend die jeweilige Funktion mehrfach zur Verfügung stellen.

Unterschiede zwischen Silberling und Elektronikmodul

Der erste und sichtbare Unterschied besteht im äußeren, also in der Bauform der Elektronikbausteine (Tab. 1).

	Silberling	Elektronik-Modul
Länge	75 mm	90 mm
Breite	45 mm	30 mm
Höhe	30 mm	16 – 105 mm (je nach Ausführung)

Tab. 1: Vergleich der äußeren Maße von Silberling und Elektronikmodul

Weiterhin besitzen die Silberlinge am Gehäuse Nut und Federschien und können auf diesem Weg direkt mit den fischertechnik-Bausteinen verbunden werden. Die Elektronikmodule sind auf eine Bodenplatte 30×90 mm von fischertechnik montiert und können auf diesem Weg direkt mit der fischertechnik-Welt verbunden werden.

Ein weiterer äußerlicher Unterschied ist das Stromversorgungssystem. Das, muss man schon sagen, wurde von fischertechnik sehr

elegant gelöst: Mit zwei Metallschienen am Gehäuse wird über einen Verbinder (wird mitgeliefert) zwischen zwei aneinander gesteckten Silberlingen die Stromversorgung verpolungssicher hergestellt. Das geht nach allen vier Seiten. Die Einspeisung erfolgt über den Gleichrichterbaustein (h4GB).

Bei den Elektronikmodulen wird die Stromversorgung über eine Kabelbrücke (71020; wird mitgeliefert) zwischen den Elektronikmodulen an der Oberseite hergestellt. Mit einer längeren Kabelbrücke (71022; 105 mm) können Elektronikmodule, die übereinander montiert sind, miteinander verbunden werden. Auf diese Weise können auch die Elektronikmodule in allen Richtungen miteinander verbunden werden. Die Stromversorgung kann direkt von einem Akku-Pack ([35537](#)) oder über ein Versorgungsmodul 70003 (entspricht dem h4GB) mit angeschlossenem Trafo, Steckernetzteil ([505287](#)) oder Power Regler ([139778](#)) versorgt werden.

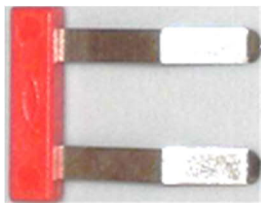


Abb. 2: Verbinder für Silberlinge



Abb. 3: Kabelbrücke für die Elektronikmodule

Für die Montage der Elektronikmodule am Modell hat man volle Bau- und Ideenfreiheit. Dafür können vielfältige Bauteile verwendet werden, z. B. Verbindungsstücke (15, 30, 45), Federnocken, Bauplatten (15×30), allerart Bausteine mit zwei Zapfen usw.

Die weiteren Unterschiede befinden sich im Inneren und betreffen die Elektronik in den Bausteinen bzw. Modulen. Tab. 2 gibt einen Überblick über die grundlegenden Unterschiede zwischen den beiden Elektronikkomponenten.

Technische Daten

Die Elektronikmodule können alle mit einer Betriebsspannung von 5 – 15 V betrieben werden. Einige Elektronikmodule können sogar bereits mit einer Betriebsspannung ab 3 V in Betrieb gehen. Höhere Spannungen können die Module beschädigen oder zerstören. Aus diesem Grund ist in den Versorgungsmodulen eine Spannungsbegrenzung enthalten, die zum Schutz der angeschlossenen Elektronikmodule bei Überschreitung der 15 V die Eingangsspannung kurzschließt.

Natürlich gibt es in den Elektronikmodulen auch analoge Elektronik. Diese kommt zum Einsatz, wenn es um Leistung auf der Ausgangsseite geht: Versorgungsmodul 70003 und 70020, Leistungsmodul 70009, Relaismodul 70018 und Inverter-Plus-Modul 70019. Zwei weitere Elektronikmodule mit analoger Elektronik sind das Messmodul 70022 und das Multifunktionsmodul 70015 – letzteres ist eigens entworfen worden, um

	Silberlinge	Elektronikmodule
Logik	negative Logik	positive Logik
Betriebsspannung	7,2 – 10,8 V=	5 – 15 V= bzw. 3 – 15 V=
Aufbau	analoge Elektronik	digitale Elektronik
Eingangsfrequenzen (max.)	2 kHz – 25 kHz	7 kHz – 10 Mhz
Anzeigen/Signalisierung	Glimmlampe 6 V	LED-Anzeige

Tab. 2: Allgemeine technische Daten von Silberlingen und Elektronikmodulen

analoge Signale aufzunehmen oder auszuwerten und in digitaler Form bereit zu stellen – inklusive einem analogen Miniverstärker für die Ausgänge.

Vergleich der Elektronikbausteine

Die Elektronikmodule wurden nicht mit dem Ziel entworfen, eine Nachbildung für die Silberlinge zu sein. So wurde z. B. positive Logik gewählt, weil jeder Computer oder Controller heutzutage mit positiver Logik arbeitet. Auf die Frage „Kann ein Elektronikmodul 1:1 ersatzweise für einen Silberling eingesetzt werden?“ muss mit einem klassischen „Jein“ geantwortet werden:

Antwort „Nein“, weil die Silberlinge mit negativer Logik arbeiten. Würde man in einer Schaltung mit mehreren Silberlingen z. B. ein h4MF herausnehmen und dafür ein 70005 Mono-Flop-Modul einsetzen, würde die Schaltung nicht funktionieren.

Würde man hingegen alle Silberlinge gegen Elektronikmodule austauschen, würde es funktionieren, weil die Silberlinge unter sich wie die Elektronikmodule die gleiche Logik verwenden. Berücksichtigt man den Unterschied und invertiert die Signale zwischen den Silberlingen und Elektronikmodulen, dann würde es ebenfalls funktionieren und somit kann man die Frage auch mit „Ja“ beantworten.

Am einfachsten wäre es, in den Modellen mit Silberlingen diese komplett durch Elektronikmodule zu ersetzen und bei den angeschlossenen Schaltern die Anschlüsse mit „+“ oder „-“ an den Schaltkontakten durch die entgegengesetzte Polarität zu ersetzen („+“ statt „-“ und „-“ statt „+“). Sollen einzelne Elektronikmodule mit Silberlingen kombiniert werden, müssen die Signalleitungen zwischen den beiden Bausteintypen invertiert werden.

Im Folgenden werde ich zu jedem Silberling das passende Elektronikmodul

vorstellen und Hinweise geben, was zu beachten ist oder welche Unterschiede es gibt, wenn das vergleichbare Elektronikmodul zum Einsatz kommen soll.

Bei der Bezeichnung der Aus- oder Eingangssignale verwende ich *low* für „-“ und *high* entsprechend für „+“.

Flip-Flop (h4FF) – JK-Flip-Flop (70007)

Der Flip-Flop-Silberling h4FF ist genau genommen ein JK-Flip-Flop und aus diesem Grund ist das JK-Flip-Flop 70007 bei den Elektronikmodulen das entsprechende Pendant.

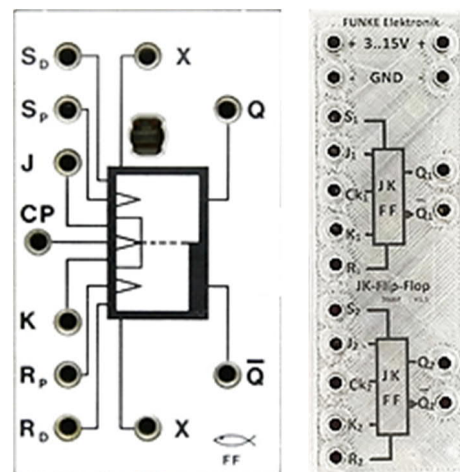


Abb. 4: h4FF (links) und JK-Flip-Flop-Modul (rechts).

Wie man in Tab. 3 sieht, müssen die Eingänge der beiden Elektronikkomponenten genau entgegengesetzt angesteuert werden (invertiertes Signal).

Die Eingänge S_P, R_P und X setzen bzw. rücksetzen das Flip-Flop (FF) über eine Flanke, wie beim CP-Eingang. Wird einer dieser Eingänge belegt, kann in der Regel dafür ersatzweise der S_X/R_X-Eingang am Elektronikmodul verwendet werden. Sollte es dennoch einmal erforderlich sein, das FF über eine Flankensteuerung anzusteuern, muss dies über den Ck_x Eingang realisiert werden. Werden in einer Schaltung beide Eingänge S_D und S_P oder R_D und R_P verwendet, müssen die Signale über ein

Funktion	Ein-/Ausgang	Aktivierung	Ein-/Ausgang	Aktivierung
	Silberling		Elektronikmodul	
Flip-Flop setzen	S _D	low	S ₁ /S ₂	high
Flip-Flop rücksetzen	R _D	low	R ₁ /R ₂	high
Sperreingang J	J	high	J ₁ /J ₂	high
Sperreingang K	K	high	K ₁ /K ₂	high
Clock	CP	high → low	Ck ₁ /Ck ₂	low → high

Tab. 3: Ansteuerung der h4FF- und Flip-Flop-Module

ODER-Gatter an den S_x- bzw. R_x-Eingang eines Flip-Flop-Elektronikmoduls (70006 / 70007 / 70008) geführt werden.

Die Elektronikmodule stellen auch andere FF-Typen zu Verfügung: Das D-Flip-Flop 70006 und das RS-Flip-Flop 70008. Werden nur die Eingänge S_D und R_D des h4FF benötigt, könnte man auch dafür das Elektronikmodul 70008 mit vier RS-FF einsetzen und damit bis zu vier h4FF Silberlinge durch ein einziges Elektronikmodul ersetzen.

Mono-Flop (h4MF) – Mono-Flop-Modul (70005)

Für den Mono-Flop-Silberling h4MF gibt es bei den Elektronikmodulen das Mono-Flop-Modul 70005.

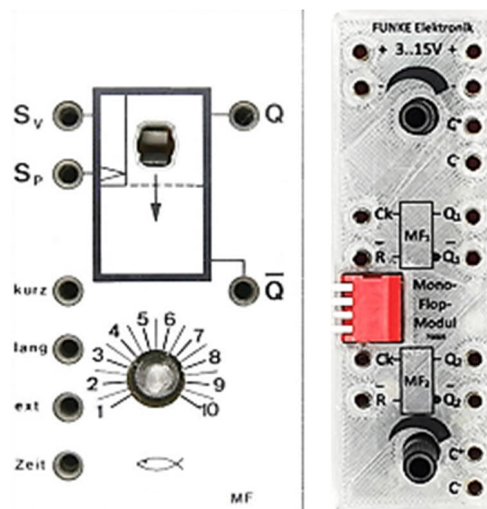


Abb. 5: h4MF (links) und Mono-Flop-Modul (rechts).

Wird die Sv Funktion in einer Schaltung benutzt, dann ist dafür ein AND-Gatter mit zwei Eingängen (70011) vor den Eingang Ck zu schalten. Der eine AND-Eingang

	Silberling	Elektronikmodul
Anzahl JK-Flip-Flops	1	2
Anzahl D -Flip-Flops	-	2
Anzahl RS-Flip-Flops	1	4
Max. Ausgangsstrom	20 mA	9 mA (bei Versorgungsspannung 9 V)
Max. Eingangsfrequenz CP bzw. Ck	2 kHz	> 10 Mhz
Silberling SD/RD	2 kHz	-
JK-FF-Modul S ₁₋₂ /R ₁₋₂	-	46 kHz
D -FF-Modul S ₁₋₂ /R ₁₋₂	-	32 kHz
RS-FF-Modul S ₁₋₄ /R ₁₋₄	-	50 kHz / 7 kHz

Tab. 4: Technischen Daten von Flip-Flop-Silberling und Flip-Flop-Modulen im Vergleich

Funktion	Ein-/Ausgang	Aktivierung	Ein-/Ausgang	Aktivierung
Silberling		Elektronikmodul		
Mono-Flop setzen	S _P	high → low	Ck	low → high
Mono-Flop sperren	S _V	high	-	
Mono-Flop rücksetzen	-		\bar{R}	low

Tab. 5: Ansteuerung von h4MF und Mono-Flop-Modul

wird mit dem Setzimpuls beschaltet und der andere AND-Eingang mit low, wenn die Funktion blockiert werden soll. Alternativ könnte auch der Reset-Eingang (R) auf low gelegt werden, um das MF zu sperren.

Das Mono-Flop 70005 verfügt intern über zwei Kondensatoren (2,2 μ F und 22 μ F), die über DIP-Schalter aktiviert werden können. Mit den beiden Kondensatoren und dem Drehregler (Potentiometer) können unterschiedliche Haltezeiten eingestellt werden. In den beiden MFs lassen sich unterschiedliche Zeitspannen einstellen. Mit extern angeschlossenen Kondensatoren können noch größere oder kleinere Zeitspannen als in Tab. 6 angegeben erzeugt werden.

Relais (h4RB) – Relaismodul (70018)

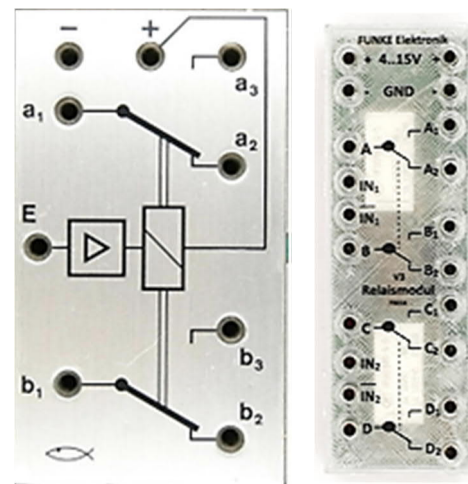


Abb. 6: h4RB (links) und Relaismodul (rechts).

Als Schalter für Motoren und Lampen war der Relaisbaustein (h4RB) gedacht. Über einen Verstärker wird ein Relais mit zwei Umschaltern (2 \times UM) angesteuert. Das Relaismodul 70018 ist ebenso aufgebaut, verfügt allerdings über zwei derartige Relais.

	Silberling	Elektronikmodul
Anzahl der Mono-Flops	1	2
Max. Ausgangsstrom	20 mA	9 mA (bei Versorgungsspannung 9 V)
Max. Eingangsfrequenz	2 kHz	> 10 Mhz
Ausgangsimpulslänge	20 ms – 30 s	200 ms – 67 s
Kleinster Kondensatorwert	keine Angaben (1)	22 pF (2)
Größter Kondensatorwert	jeder Wert über 47 μ F (1)	jeder Wert über 22 μ F (2)

(1) externer Kondensator zwischen Buchsen „Zeit“ und „ext“

(2) externer Kondensator zwischen Buchsen „C-“ und „C+“

Tab. 6: Technischen Daten im Vergleich zwischen Mono-Flop Silberling und Mono-Flop-Modul

Funktion	Ein-/Ausgang	Aktivierung	Ein-/Ausgang	Aktivierung
Silberling		Elektronikmodul		
Relais anziehen	E	low	IN *	high
Relais anziehen	---		$\overline{\text{IN}}$ *	low

* Die beiden Eingänge sind über ein OR-Gatter verknüpft.

Tab. 7: Ansteuerung von h4RB und Relaismodul.

Beim Relaismodul 70018 können die Relais mit low oder high geschaltet werden. Somit kann das Relaismodul direkt den h4RB in Schaltungen ersetzen.

Alternativ kann noch das Inverter-Plus-Modul 70019 mit vier Einschaltern verwendet werden, wenn es darum geht, kleine Verbraucher bis 230 mA zu betreiben. Hierbei handelt es sich um Einschalter, die auf einer Seite fest mit high verbunden sind. Dafür kann der Einschalter mit einer Frequenz über 10 MHz betrieben werden.

Für größere Schaltleistungen bis 2,3 A steht das Leistungsmodul 70009 mit einem Umschalter zur Verfügung, wobei die Schaltkontakte mit high und low fest verbunden sind. Bei dem Leistungsmodul können ebenfalls Schaltvorgänge mit über 10 Mhz erfolgen.

Bei beiden Elektronikmodulen können die Schalter mit low oder mit high aktiviert werden.

Grundbaustein (h4G) – Multifunktionsmodul (70015)

Der Grundbaustein ist ein Differenzverstärker, der in erster Linie zum Erfassen

und Auswerten von analogen Signalen eingesetzt wird, wie Feuchtigkeitsfühler, Temperaturfühler (PTC, NTC) und Helligkeitssensoren (Fotozelle). Das Multifunktionsmodul stellt die gleichen Funktionalitäten zur Verfügung und kann damit ebenso direkt für den Grundbaustein eingesetzt werden.

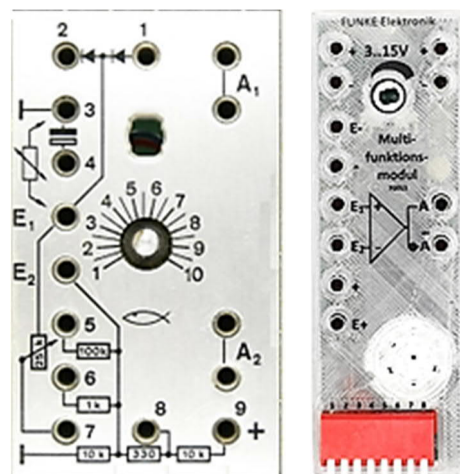


Abb. 7: h4GB (links) und Multifunktionsmodul (rechts).

Die Ansteuerung ist zwischen Grundbaustein und dem Multifunktions-Modul im Wesentlichen identisch. Es gibt die gleichen Eingänge (E_1 , E_2) und die gleichen Ausgänge ($A_1 = A$; $A_2 = \bar{A}$).

	Silberling	Elektronikmodul
Anzahl Relais	1	2
Anzahl der Schalter pro Relais	2 x UM	2 x UM
Minimale Schaltzeit	20 ms	5 ms
Schaltleistung max.	40 W	90 W
Schaltstrom max.	1 A (40 V)	3 A (30 V)

Tab. 8: Technischen Daten im Vergleich zwischen Relaisbaustein und Relaismodul.

	Silberling	Elektronikmodul
Anzahl Differenzverstärker	1	1
Max. Eingangsfrequenz	keine Angaben	> 10 Mhz
Max. Ausgangsstrom	20 mA	87 mA

Tab. 9: Technischen Daten im Vergleich zwischen Grundbaustein und Multifunktionsmodul.

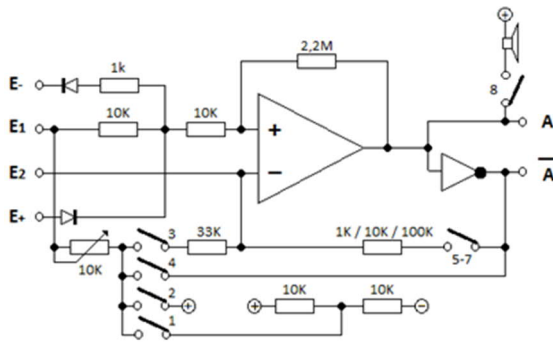


Abb. 8: Interne Beschaltung des Differenzverstärkers.

Das Multifunktionsmodul bietet ein paar Vorteile gegenüber dem Grundbaustein. Es brauchen keine Kabelbrücken gesteckt zu werden, sondern die Brücken werden mit kleinen DIP-Schaltern hergestellt. Die Ausgänge verfügen über einen Mini-Verstärker und bei Bedarf kann ein Ausgang mit einem internen Lautsprecher verbunden werden.

Der DIP-Schalter 3 dient Messungen von kleinsten Spannungen zwischen den Eingängen E_1 und E_2 z. B. bei Spulen, in denen eine Spannung durch einen Magneten induziert wird oder durch Schallwellen von einem Mikrofon. DIP-Schalter 4 ist für Rückkopplungen zum Ausgang.

Silberling	Elektronikmodul
Buchse 1	Buchse E+
Buchse 2	Buchse E-
Buchse 3	„-“ (GND)
Buchse 4	-
Buchse E_1	Buchse E_1
Buchse E_2	Buchse E_2
Buchse 5	DIP-Schalter 7
-	DIP-Schalter 6

Silberling	Elektronikmodul
Buchse 6	DIP-Schalter 5
Buchse 7	---
Buchse 8	DIP-Schalter 1
Buchse 9 (+)	DIP-Schalter 2 (+)
Buchsen A_1	Buchse A
Buchsen A_2	Buchse \bar{A}
---	DIP-Schalter 3

Tab. 10: Vergleich der Anschlüsse des h4GB mit den DIP-Schaltern des Multifunktionsmoduls

AND-NAND (h4AN) – AND-NAND-Modul (70010, 70011)

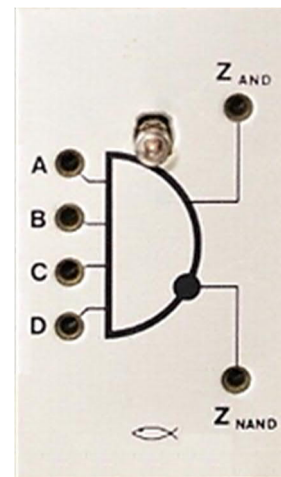


Abb. 9: h4AN

Der h4AN kann als 2er, 3er oder 4er AND-NAND eingesetzt werden. Bei den Elektronikmodulen gibt es dafür ein AND-NAND-Modul mit 4 Eingängen (70010) und mit 2 Eingängen (70011).

Funktion	Ein-/Ausgang	Aktivierung	Ein-/Ausgang	Aktivierung
	Silberling		Elektronikmodul	
AND setzen (Eingänge)	A & B & C & D	low	A _x & B _x & C _x & D _x	high
	A & B	low	A _x & B _x	high
AND gesetzt (Ausgang)	Z _{AND}	low	Y _x	high
NAND gesetzt (Ausgang)	Z _{NAND}	high	$\overline{Y_x}$	low

Tab. 11: Ansteuerung von h4AN und AND-NAND-Modul.

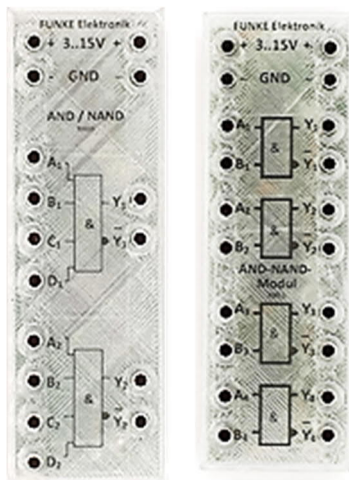


Abb. 10: AND-NAND-Module mit vier (links) und zwei (rechts) Eingängen

OR-NOR (h4ON) – OR-NOR-Modul (70012, 70013)

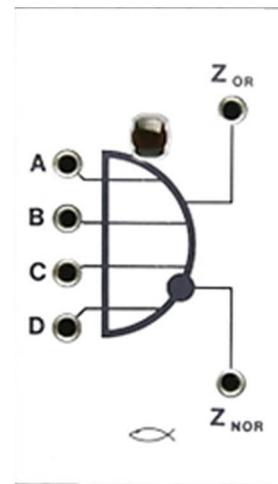


Abb. 11: h4ON

Der h4ON kann als 2er-, 3er- oder 4er-OR-NOR eingesetzt werden. Bei den Elektronikmodulen gibt es dafür ein OR-NOR-Modul mit 4 Eingängen (70012) und mit 2 Eingängen (70013).

	Silberling	Elektronikmodul 4er AND-NAND	Elektronikmodul 2er AND-NAND
Anzahl der Gatter	1	2	4
Max. Eingangsfrequenz	15 kHz	> 10 Mhz	> 10 Mhz
Max. Ausgangsstrom	20 mA	9 mA *	9 mA *

* bei Versorgungsspannung 9 V

Tab. 12: Technische Daten im Vergleich zwischen AND-NAND Silberling und AND-NAND-Modul

Funktion	Ein-/Ausgang	Aktivierung	Ein-/Ausgang	Aktivierung
	Silberling		Elektronikmodul	
OR setzen (Eingänge)	$A \wedge B \wedge C \wedge D$	low	$A_x \wedge B_x \wedge C_x \wedge D_x$	high
	$A \wedge B$	low	$A_x \wedge B_x$	high
OR gesetzt (Ausgang)	Z_{OR}	low	Y_x	high
OR gesetzt (Ausgang)	Z_{NOR}	high	$\overline{Y_x}$	low

Tab. 13: Ansteuerung von h4ON und OR-NOR-Modul

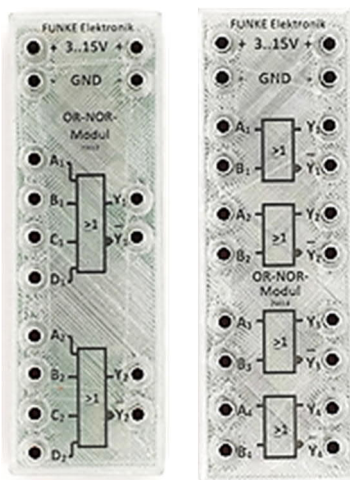


Abb. 12: OR-NOR-Module mit vier (links) und zwei (rechts) Eingängen

Mikrofon-Lautsprecher-Baustein (h4ML) – Multifunktionsmodul (70015)

Für den Mikrofon-Lautsprecher-Baustein gibt kein eigenes Modul, sondern im Multifunktionsmodul ist ein Mini-Lautsprecher mit integriert und somit wird kein separates Modul benötigt.

Soll ein Mikrofon zum Einsatz kommen, dann kann ein handelsübliches Mikrofon an

die Anschlüsse E1 und E2 des Multifunktionsmoduls angeschlossen werden.

Gleichrichterbaustein (h4GB) – Versorgungsmodul (70003, 70020)

Der Gleichrichterbaustein kann Wechselspannungen und pulsierende Gleichspannungen glätten und für die Elektronik brauchbar machen. Die gleiche Funktion übernimmt das Versorgungsmodul 70003 und ist damit 1:1 austauschbar mit dem h4GB.



Abb. 13: h4GB (links) und Versorgungsmodul (rechts)

	Silberling	Elektronikmodul 4er OR-NOR	Elektronikmodul 2er OR-NOR
Anzahl der Gatter	1	2	4
Max. Eingangsfrequenz	15 kHz	> 10 Mhz	> 10 Mhz
Max. Ausgangsstrom	20 mA	9 mA *	9 mA *

* bei Versorgungsspannung 9 V

Tab. 14: Technische Daten im Vergleich zwischen OR-NOR Silberling und OR-NOR-Modul

Das Versorgungsmodul 70003 verfügt zusätzlich über eine Niederspannungsanzeige für Akku-Betrieb (variabel einstellbar), eine Anschlussbuchse für das Schaltnetzteil [505287](#), Schraubklemmen für einen Anschluss über Drähte und einen Überspannungsschutz, der bei einer Spannung über 15 V die Eingangsseite kurzschließt.

Das Versorgungsmodul 70020 hat die gleichen Zusätze wie das Versorgungsmodul 70003, ist jedoch für Speisespannungen von über 15 V ausgelegt und wandelt diese in eine Gleichspannung von 9,2 - 9,6 V um.

Dyn.-AND (h4DA)

Für den Dyn.-AND (h4DA) gibt es kein vergleichbares Elektronikmodul. Dieser Baustein stellt eine Sperrfunktion für wechselnde Signale zur Verfügung, die über ein AND-Gatter realisiert wird. Schaut man sich den Spannungsverlauf am Ausgang des Dyn.-ANDs an (Abb. 11), dann zeigt dieser einen negativen Spannungsaus Schlag; derartige negative Spannungsaus schläge würden digitale Bausteine zerstören.

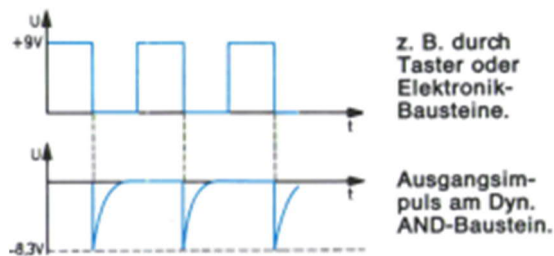


Abb. 14: Negativer Impuls von einem dynamischen AND-Baustein erzeugt

Um ein Dyn.-AND Baustein zu ersetzen, kann ein Mono-Flop eingesetzt werden, wobei das Eingangssignal invertiert werden muss, wenn die fallende Flanke der Auslöser sein soll. Geht es nur darum, einen

Impuls zu sperren, kann auch ein normales 2er-AND-Gatter verwendet werden, wobei ein Eingang mit dem Signal beschaltet wird, während der andere Eingang zum Sperren verwendet wird.

Weitere Bausteine

Die „hobbylabor 1“-hl-Bausteine sind für experimentelle Zwecke gedacht, etwa der hl 1 PB Potentiometerbaustein oder das hl 1 EF Experimentierfeld. Bei den Elektronikmodulen gibt es das Messmodul, das ein V-A-Meter und ein frei verwendbares Potentiometer von 1 kΩ enthält. Für das Experimentierfeld gibt es vergleichsweise das Experimentiermodul, das zusätzlich ein Set aus neun Widerständen, zwei Dioden und zwei Steckern (mit Schrauben) enthält, sowie eine Tabelle zur Bestimmung von Widerständen.

Wer weitere spezielle Fragen zu den Elektronikmodulen oder zu Schaltungen mit Silberlingen hat, kann diese per E-Mail an mich (elektronik@funke4you.de) richten. Gerne bin ich behilflich bei der Lösungsfindung eines Problems. Informationen bekommt man auch bei Franz Santjohanser [1], wo die Elektronikmodule, die Zusatzmodule, Zubehör und Labore erhältlich sind.

Quellenangaben

- [1] santjohanser: [Spielen. Lernen. Technik.](#) Autorisierter fischertechnik-Fachhändler.
- [2] Fischerwerke: [Leitfaden für die fischertechnik-Elektronik-Bausteine.](#) In der [ft-Datenbank](#).

	h4DA			Mono-Flop-Modul			AND-Gatter		
B _p	A _v	Z	Ck	R	Q	A	B	Y	
high → low	low	0-1-0	low → high	high	0-1-0	low → high	high	0-1	
	high	0		low	0		low	0	

Tab. 15: Ansteuerung von h4DA, eines Mono-Flop-Modul und eines normalen AND-Gatters

Elektronik

Silberlinge: Flip-Flop in Hardware oder Software?

Peter Krijnen

Im Forum gibt es einen Thread, in dem die Funktionsweise des Flip-Flops diskutiert wird [1], insbesondere wenn die Lampe durch eine LED ersetzt wird. Eine Person hat sich die Mühe gemacht, einige Silberlinge zu simulieren. Dazu nutzte er das Programm LTspice [2]. Dieses Programm kann kostenlos heruntergeladen und verwendet werden.

Ich selbst habe in der Vergangenheit schon mit LTspice gearbeitet, daher ist mir dieses Programm bekannt. Dass es in diesem Thread um das Flip-Flop geht, war für mich der Anlass, das Flip-Flop selbst in LTspice zu simulieren. Es hat einige Zeit gedauert, bis ich das hinbekommen habe, aber am Ende ist es mir gelungen.

Aber wie sieht es in der Realität aus? Dazu müssen wir uns zunächst die Spannungsversorgung anschauen. Ich rate daher jedem, Teil 2 [3] noch einmal zu lesen.

Anhand mehrerer Abbildungen möchte ich zu erklären versuchen, was im Gleichrichtermodul und im Flip-Flop passiert, wenn die Stromversorgung angeschlossen wird.

Um die Messungen richtig durchführen zu können, habe ich einen Messaufbau aufgebaut, der aus folgenden Teilen besteht (Abb. 1):

- einem Transformator 814 für die Wechselspannung (AC),
- einem ELV Switch-Power-Supply SPS 7330 für die Gleichspannung (DC),
- einem Standard-Gleichrichterbaustein (h4GB),
- einem mit einem 7809-Stabilisator modifizierten Gleichrichterbaustein (h4GB + 7809),

- einem Standard-Flip-Flop (h4FF),
- einem Potentiometerbaustein (PB),
- einem kleinen Breadboard,
- einer LED und
- einem Schalter zum Kurzschließen des GB.

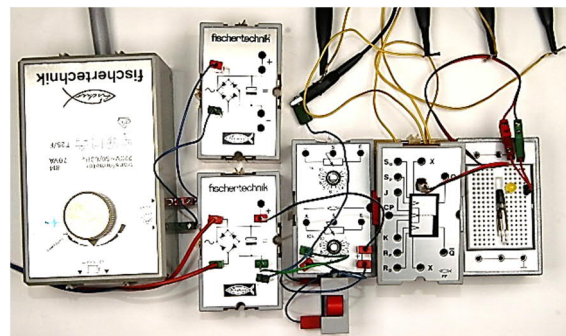


Abb. 1: Der Messaufbau

Das FF wird nicht direkt aus dem GB gespeist, sondern über den PB. Dies kann aber auch ein Relais-Baustein oder ein Verstärker-Baustein sein. Allerdings haben RB und VB eingebaute Bauteile, die die Versorgungsspannung belasten. Optional können zwei Drähte direkt an den Kontaktfedern an der Seite des FF-Gehäuses angeklemt werden.

Untersuchungen

Vorbereitung

Abb. 2 zeigt, was passiert, wenn die Wechselspannung vom Trafo 814 an den

Eingang des GB angelegt wird: Durch die gleichgerichtete Wechselspannung wird der Kondensator alle 10 ms aufgeladen. Erst nach etwa 100 ms ist der Kondensator so weit aufgeladen, dass sich ein einigermaßen stabiler Pegel von 11,4 V einstellt. Eine kleine Welle bleibt jedoch.

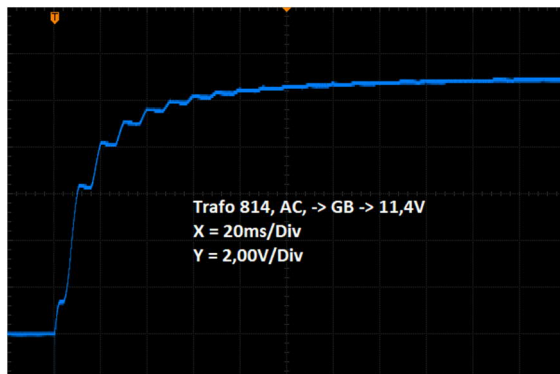


Abb. 2

Für Abb. 3 habe ich den Trafo 814 durch das SPS 7330 ersetzt. Bei 12,3 V DC am Eingang des GB werden nun auch 11,4 V ausgegeben. Wir sehen nun, dass, da die angeschlossene Gleichspannung nicht mehr gleichgerichtet werden muss, die Spannung bereits nach ca. 8 ms das Maximum erreicht. Erst dann wird der Kondensator wieder auf ein stabiles Niveau von 11,4 V aufgeladen.

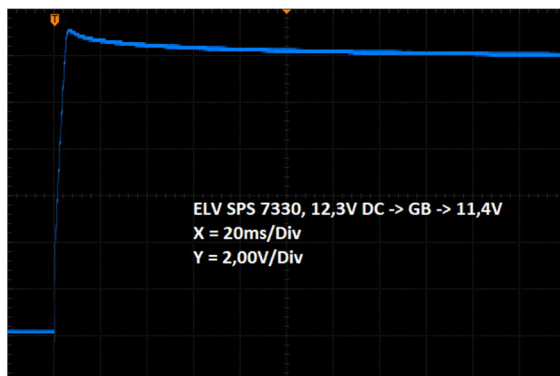


Abb. 3

In Abb. 4 sehen wir das gleiche wie auf Abb. 2. Wieder wird der Trafo 814 benutzt, aber ich habe den GB jetzt mit einem 7809-Stabilisator ausgestattet. Da der Platz für den 2200 μ F Kondensator nicht mehr ausreichte, habe ich ihn durch einen kleineren mit 1000 μ F ersetzt. Dem Ausgang wird

dann ein 10- μ F-Kondensator hinzugefügt. Dies soll die Stabilität des 7809 verbessern.

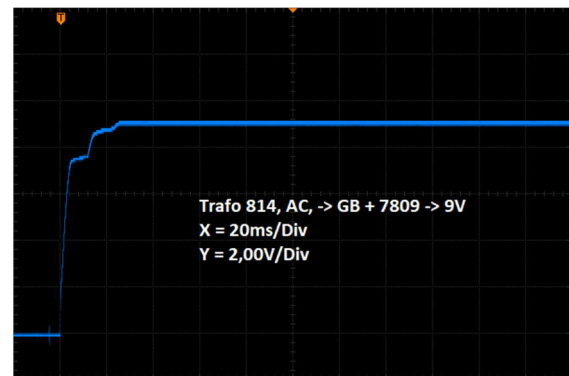


Abb. 4

Wir sehen wieder alle 10 ms einen „Wellen-top“, der den Kondensator auflädt. Nach dem dritten „top“, innerhalb von 30 ms, liegt der Ausgang sofort stabil bei 9 V.

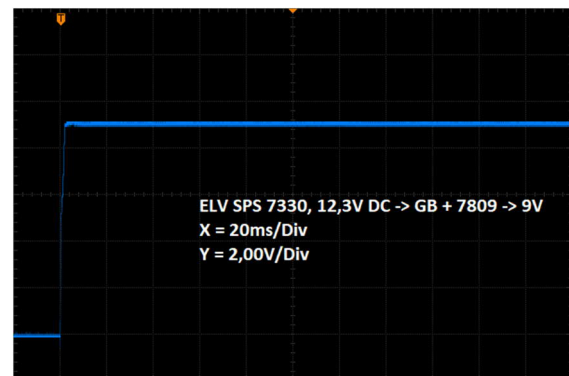


Abb. 5

Abb. 5: Wie in Abb. 4, aber wieder mit 12,3 V DC von der SPS. Die Spannung ist jetzt nach ca. 5 ms stabil.

Aus den oben genannten Bildern können wir schließen, dass, wenn bereits eine stabile Gleichspannung am Eingang des GB anliegt, die Einschalteneffekte reduziert werden. Wird der GB zusätzlich noch mit einem Stabilisierungs-IC ausgestattet, sind die Auswirkungen schon minimal.

Jetzt, da wir das wissen, stellt sich die Frage: Was passiert im Flip-Flop?

Schalten am Eingang des GB, Trafo 814 (AC) Zeitskala 20 ms/div

Für die nächsten Bilder habe ich wieder den Trafo 814 und den Standard-GB verwendet.

Um zu verstehen, was wir sehen:

- Kanal 1 = Kollektor T1 (gelb)
- Kanal 2 = Basis T1 (cyan)
- Kanal 3 = Kollektor T2 (violett)
- Kanal 4 = Basis T2 (blau).

Ich habe die Zeitbasis auf 20 ms eingestellt.

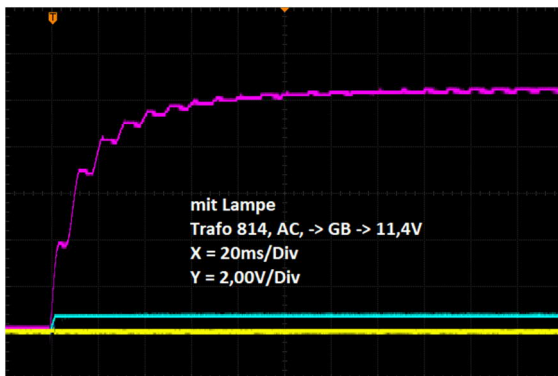


Abb. 6: Mit Lampe

Abb. 6 zeigt, was im FF passiert, wenn wir die Spannung an den GB anschließen. Der FF ist mit einer Lampe ausgestattet. Wir sehen hier dasselbe wie in Abb. 3. Wir sehen, dass es wieder etwa 100 ms dauert, bis der Pegel am Kollektor von T2 stabil ist. Innerhalb von etwa 5 ms wird die Basis von T1 „high“, was bedeutet, dass der Kollektor von T1 auf „0“ gezogen wird. Die Basis von T2 ist „low“, sie sperrt und der Pegel am Kollektor von T2 ist daher „high“ und die Lampe brennt nicht.

Abb. 7: Wie Abb. 6, aber jetzt ohne Lampe. Innerhalb von etwa 10 ms wird die Basis von T2 „high“, was bedeutet, dass der Kollektor von T2 auf „0“ gezogen wird. Die Basis von T1 ist „low“, sie ist gesperrt und der Pegel am Kollektor von T1 ist daher „high“.

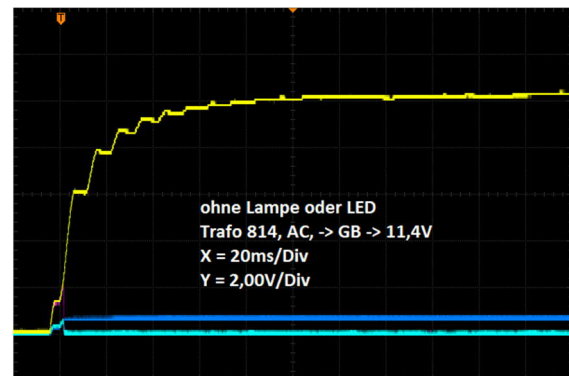


Abb. 7: Ohne Lampe oder LED

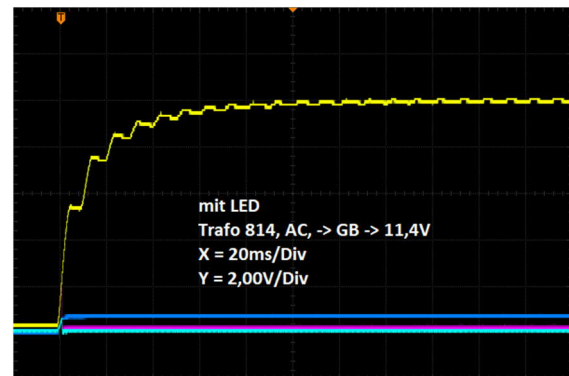


Abb. 8: Mit LED

Abb. 8: Langsam wird es etwas eintönig. Wie in Abb. 7, jetzt mit einer LED verbunden. Innerhalb von etwa 5 ms wird die Basis von T2 „hoch“, was bedeutet, dass der Kollektor von T2 auf „0“ gezogen wird. Die Basis von T1 ist „low“, sie ist gesperrt und der Pegel am Kollektor von T1 ist daher „high“. Die LED leuchtet.

Schalten am Ausgang des GB Trafo 814 (AC) Zeitskala 20 ms/div

Bei den oben beschriebenen Messungen habe ich jeweils den Eingang des GB abgeklemmt und den Kondensator durch Kurzschließen des Ausgangs entladen. Bei den folgenden Messungen habe ich immer den Ausgang des GB verbunden. Dazu darf keine direkte elektrische Verbindung zum FF bestehen. Deshalb habe ich den GB vom FF getrennt und über den PB die Verbindung zum FF hergestellt.

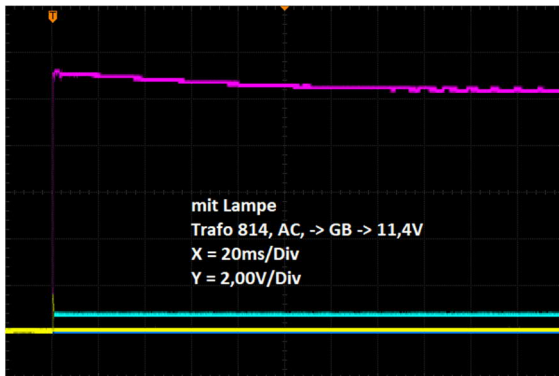


Abb. 9: Mit Lampe

In Abb. 9 ist das gleiche zu sehen wie in Abb. 6. Allerdings liegt die Versorgungsspannung bereits nach wenigen μs voll an. Und die Lampe brennt nicht.

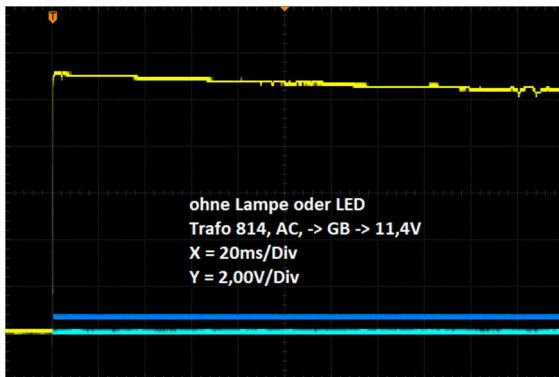


Abb. 10: Ohne Lampe oder LED

Wie zu erwarten, ist Abb. 10 gleich Abb. 7. Aber auch jetzt ist die Versorgungsspannung schon nach wenigen μs voll vorhanden.

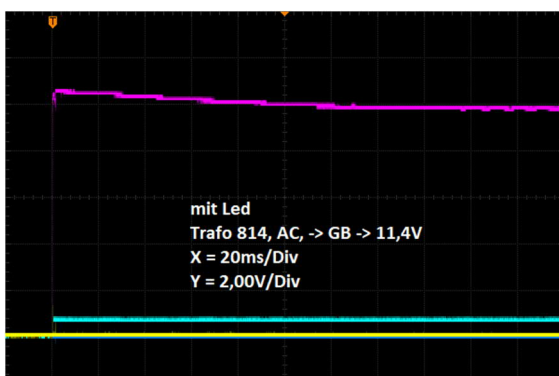


Abb. 11: Mit LED

Jetzt wird man vielleicht denken, dass Abb. 11 gleich Abb. 8 sein müsste. Dem ist aber nicht so. Dafür gibt es drei Gründe: Zum einen liegt der Widerstand der Lampe

zwischen 35 und 50 Ω . Zweitens hat die LED am Anfang einen viel höheren Widerstand, viel höher als die 560 Ω von R7. Drittens ist die Ladezeit des Kondensators im GB viel zu lang. Da die Ladezeit des Kondensators wegfällt, steht die Spannung nun sofort zur Verfügung, wodurch die Transistoren wesentlich früher schalten können. Die LED leuchtet nicht.

Schalten am Eingang des GB SPS 7330 (DC) Zeitskala 20 $\mu\text{s}/\text{div}$

Die Änderungen für die folgenden Messungen sind der SPS 7330 für die Gleichspannung und der modifizierte GB mit 9 V-Stabilisator. Ich habe die Zeitbasis auf 200 μs eingestellt, und ich schalte die Stromversorgung am Eingang des GB ein.

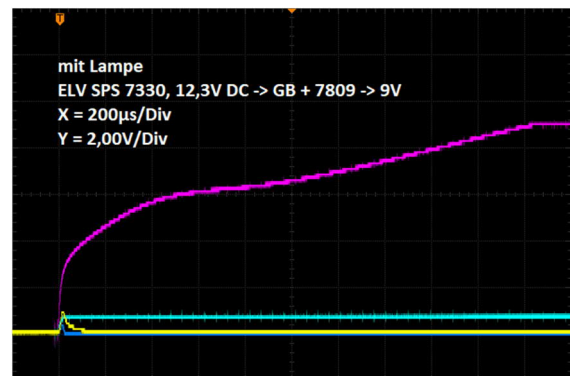


Abb. 12: Mit Lampe

Abb. 12: Es dauert 2 ms, bis sich der Pegel am Kollektor von T2 stabilisiert hat. Der eigentliche Schaltmoment ist bereits nach ca. 10 μs . Die Lampe brennt nicht.

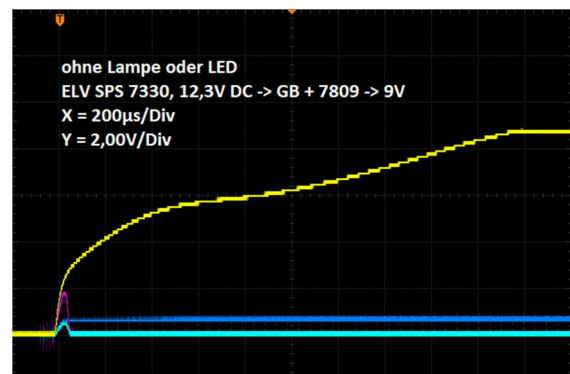


Abb. 13: Ohne Lampe oder LED

Abb. 13: wieder ohne Lampe.

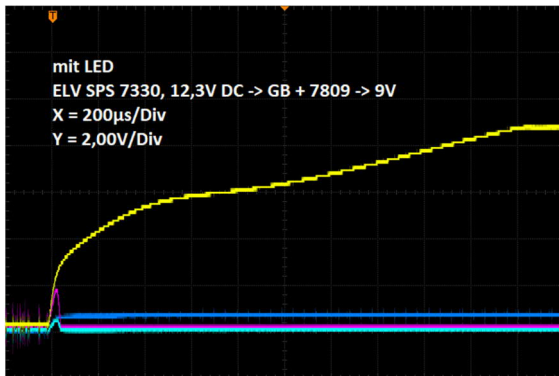


Abb. 14: Mit LED

Abb. 14: Obwohl der eigentliche Schaltmoment hier bereits nach ca. $10\ \mu\text{s}$ stattgefunden hat, ist klar, dass die benötigte Zeit zu lang ist und daher T2 statt T1 geschaltet wird. Die LED leuchtet daher auf.

Schalten am Ausgang des GB SPS 7330 (DC) Zeitskala $10\ \mu\text{s}/\text{div}$

Bei den letzten drei Bildern schalte ich den Ausgang des GB wieder um. Die Zeitbasis ist jetzt auf $10\ \mu\text{s}$ eingestellt.

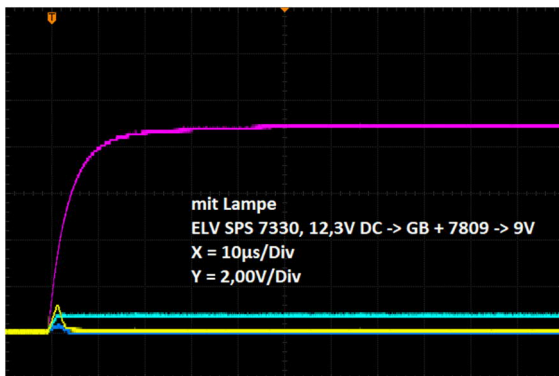


Abb. 15: Mit Lampe

Abb. 15 zeigt den Verlauf der Spannung am Kollektor von T2 (violette Linie). Nach etwa $50\ \mu\text{s}$ ist die Spannung stabil. Der eigentliche Schaltmoment liegt bei ca. $3\ \mu\text{s}$, und die Lampe brennt nicht.

Abb. 16: Ohne Lampe dauert es länger, etwa $65\ \mu\text{s}$, bis der Pegel am Kollektor von T1 (gelbe Linie) stabil ist. Der eigentliche Schaltmoment liegt bei ca. $6\ \mu\text{s}$.

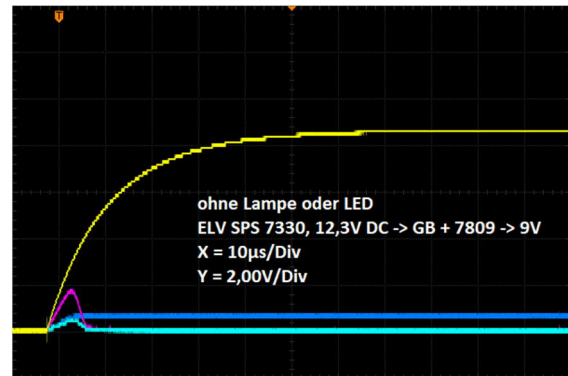


Abb. 16: Ohne Lampe oder LED

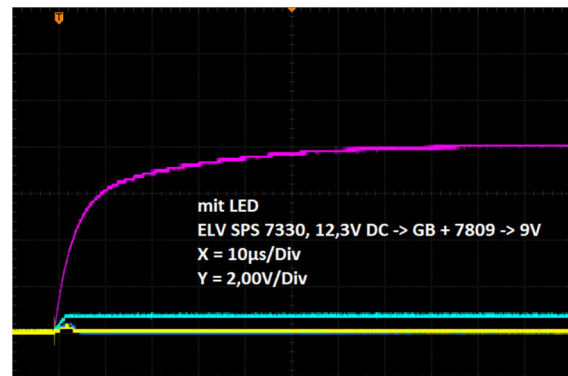


Abb. 17: Mit LED

Abb. 17 zeigt, dass die Schaltung mit LED der mit Lampe gleicht. Der eigentliche Schaltmoment liegt wieder bei ca. $3\ \mu\text{s}$ und die LED leuchtet nicht. Der Unterschied besteht jedoch darin, dass dieses Schaltmoment weniger deutlich ist (gelbe Linie). Es dauert auch viel länger, etwa $85\ \mu\text{s}$, bis die Spannung am Kollektor von T2 stabil ist. Da die LED auch einen viel größeren Widerstand hat als die Lampe, ist auch die Spannung am Kollektor von T2 geringer: $8,8\ \text{V}$ bei Lampe und $8\ \text{V}$ bei der LED.

Je nachdem, welche LED verwendet wird, muss der Vorwiderstand angepasst werden, damit der Strom durch die LED begrenzt wird. Ansonsten würde ein Strom von $28\ \text{mA}$ durch die LED fließen. Dies hängt, wie erwähnt, von der verwendeten LED ab. Ein Wert von $1000\ \Omega$ für R19 erscheint mir ausreichend. Da die von mir verwendeten LEDs viel Licht erzeugen, habe ich den Wert von R19 auf $2700\ \Omega$ erhöht.

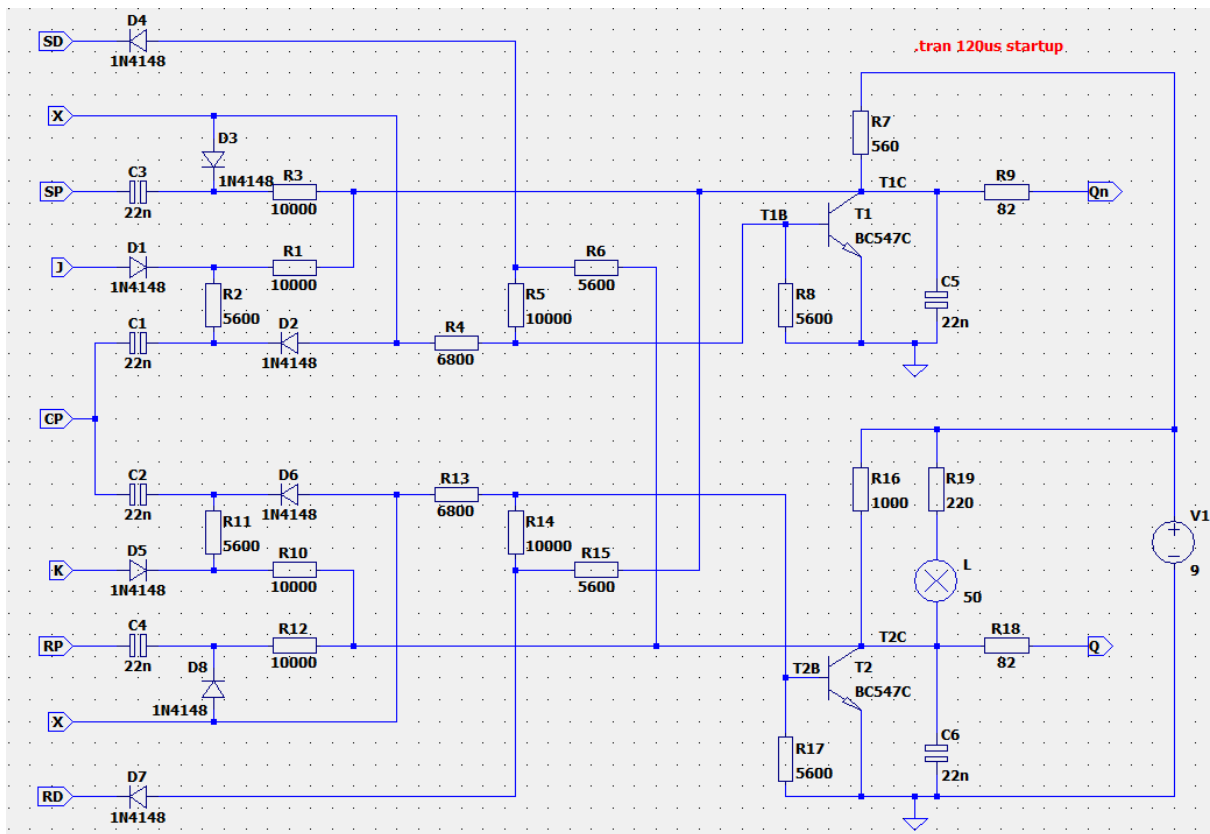


Abb. 18: Schaltplan in LTspice

Da es nicht praktikabel ist, immer zuerst den GB an die Spannungsversorgung anzuschließen und erst dann den Ausgang mit den anderen Silberlingen zu verbinden, ist es besser, R16 zu verkleinern. Um den normalen Betrieb wiederherzustellen, habe ich für meine FFs für R16 einen Wert von 560 Ω gewählt.

LTspice

Wie sieht es aus, wenn wir das Flip-Flop in LTspice [2] simulieren? Das Problem ist jedoch, dass in der Datenbank keine Lampe verfügbar ist. Es gibt eine NEON-Lampe, aber die funktioniert anders als die fischer-technik-Lampe. Es ist auch nicht möglich, die Eingänge während der Simulation zu ändern. Es ist auch kein Schalter oder Taster in der Datenbank vorhanden. Deshalb habe ich selbst eine Lampe gezeichnet.

Nachdem ich das Schaltbild (Abb. 18) in LTspice gezeichnet hatte, konnte ich die

Simulation starten. Zuerst musste ich die „Traces“ auswählen:

- Kanal 1: $V(t1c)$ (gelb)
- Kanal 2: $V(t1b)$ (cyan)
- Kanal 3: $V(t2c)$ (violett)
- Kanal 4: $V(t2b)$ (blau)

Siehe auch die Beschriftungen am oberen Rand von Abb. 19. Da die „RUNTIME“ standardmäßig auf 1 Sekunde eingestellt ist, war es notwendig, die „Transient“-Einstellungen auf „tran 120 μ s startup“ zu setzen, damit die Anzeige mit der des Oszilloskops übereinstimmt. Leider musste ich die vertikale Skalierung immer manuell einstellen, da sich das „Auto-Scaling“ nicht abschalten lässt.

Um das Flip-Flop in LTspice zu simulieren, habe ich anstelle einer Lampe einfach einen 50 Ω -Widerstand verwendet. Um den Betrieb mit einer LED zu bekommen, habe ich R19 auf 1000 Ω erhöht und verschiedene LEDs aus der Datenbank ausprobiert.

Indem $R16$ dann gleich $R7 = 560 \Omega$ gemacht wird, sind die Verhältnisse wieder normal (Abb. 23).

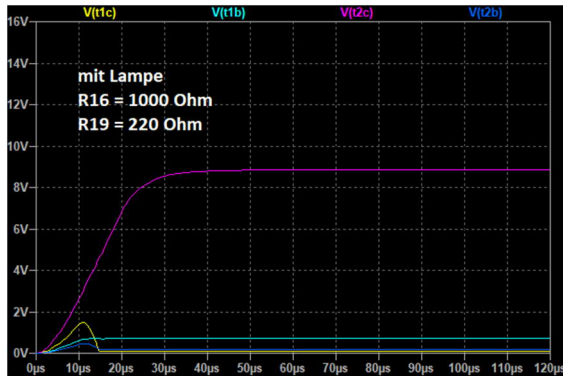


Abb. 19

Abb. 19 zeigt noch einmal den Spannungsverlauf mit der Lampe. Das sollte Abb. 15 entsprechen, oder? Dass dem nicht so ist, liegt einfach daran, dass in der simulierten Welt alles ideal und ohne Abweichungen ist.

Wir sehen deutlich, dass die Spannung am Kollektor von T2 (violett) schnell ansteigt und nach ca. $40 \mu\text{s}$ stabil bei ca. $8,8 \text{ V}$ liegt. Dadurch steigt auch die Spannung an der Basis von T1 (cyan). Nach ca. $12 \mu\text{s}$ schaltet es durch. Dies ist deutlich an der Höhe der Spannung am Kollektor von T1 (gelb) zu erkennen: Er wird auf „0“ gezogen.

Die imaginäre Lampe konnte also nicht brennen.

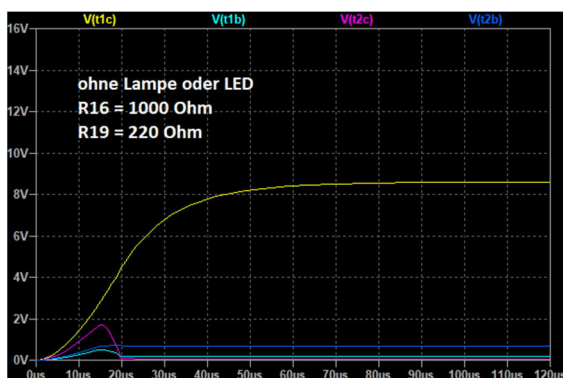


Abb. 20

Abb. 20 zeigt den Verlauf, wenn wir die Lampe weglassen. Wir sehen nun, dass die Spannung am Kollektor von T1 (gelb) schnell ansteigt, aber erst nach ca. $70 \mu\text{s}$

stabil bei ca. $8,5 \text{ V}$ liegt. Dadurch steigt auch die Spannung an der Basis von T2 (blau). Nach etwa $16 \mu\text{s}$ schaltet er durch. Dies ist deutlich an der Höhe der Spannung am Kollektor von T2 (violett) zu erkennen: Sie wird jetzt auf „0“ gezogen.

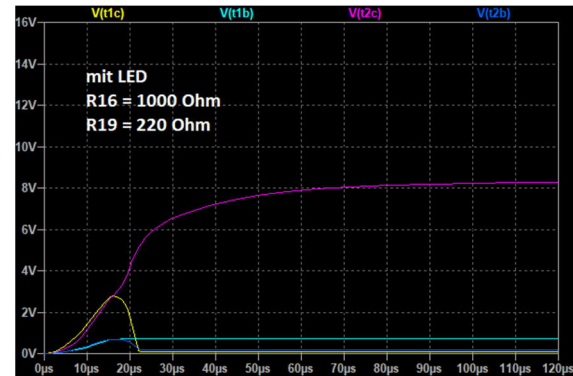


Abb. 21

Abb. 21: Die Lampe wurde durch eine LED, Typ NSPW500BS der Marke Nichia, ersetzt. Ich selbst hatte ein ähnliches Bild erwartet, wie es in Abb. 17 zu sehen ist. Wenn ihr eine andere LED aus der Datenbank auswählt, wird dieses Bild aber immer anders sein. Es kommt also auf die Eigenschaften der gewählten LED an. Der Widerstand der LED ist am Anfang um ein Vielfaches höher als der einer Lampe.

Wir sehen, dass die Spannung am Kollektor von T1 etwas schneller ansteigt als die an T2. Der Kippunkt liegt bei etwa $17 \mu\text{s}$: Dann sind beide Spannungen kurzzeitig gleich, die Spannung an T2 steigt dann aber schnell auf $8,3 \text{ V}$ an. Allerdings dauert es jetzt $90 \mu\text{s}$, bis dies der Fall ist. Unsere virtuelle LED wird nicht aufleuchten können.

Auffällig ist, dass die Spannungshöhe des Kippunktes knapp 3 V beträgt. Mit Lampe sind es $1,7 \text{ V}$ und ohne Lampe $1,8 \text{ V}$.

Abb. 22: Im Gegensatz zu Abb. 21 ist jetzt $R19 = 1000 \Omega$. Dieses Bild ist fast identisch mit Abb. 20. Der Kippunkt liegt immer noch bei $16 \mu\text{s}$, aber er ist etwas höher: $2,1 \text{ V}$. In diesem Fall leuchtet die virtuelle LED auf.

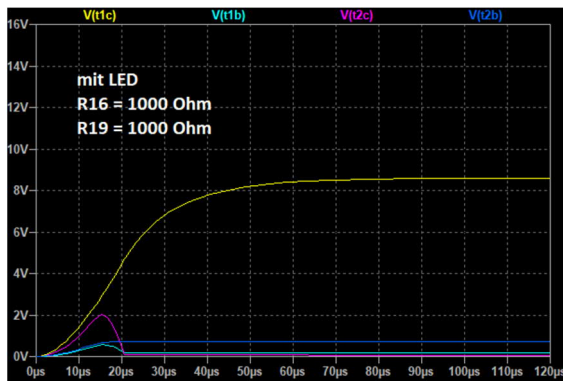


Abb. 22

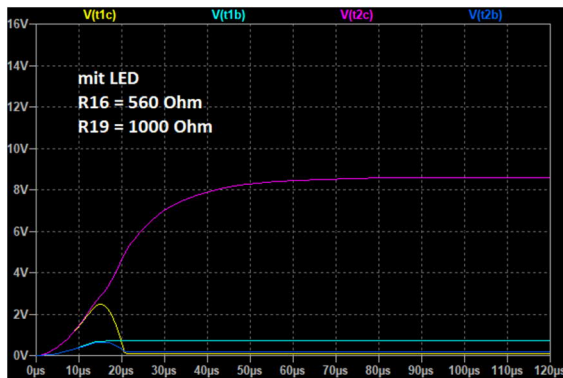


Abb. 23

Die Darstellung in Abb. 23 hätten wir schon in Abb. 21 sehen sollen. Das Bild zeigt, dass R16 auf 560 Ω abgesenkt wurde.

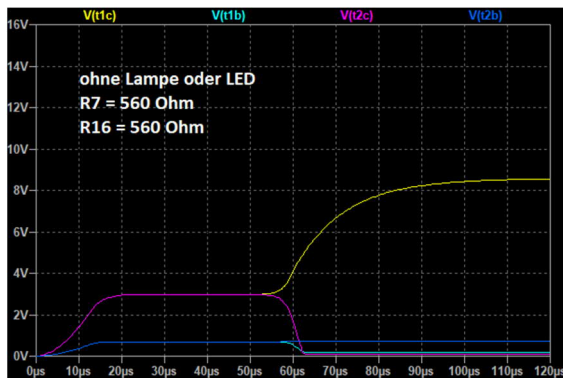


Abb. 24

Der Kipppunkt liegt immer noch bei 16 μ s, aber jetzt bei etwa 2,5 V. Ergebnis: keine leuchtende LED.

Aber was würde passieren, wenn $R16 = R7 = 560 \Omega$ gemacht würde? Und ohne Lampe oder LED. Das sehen wir in Abb. 24.

Nach 22 μ s beginnt eine Art „stabile“ instabile Phase, die erst nach 30 μ s in eine der

beiden stabilen Lagen übergeht. Jedes Mal, wenn die Simulation startet, bekomme ich dieses Bild. Dies gilt natürlich nur in einer idealen Welt, in der die verwendeten Bauteile keine Toleranzen aufweisen. In der realen Welt, in der wir alle leben, sind wir aber abhängig von den Abweichungen, die die Bauteile nun mal haben.

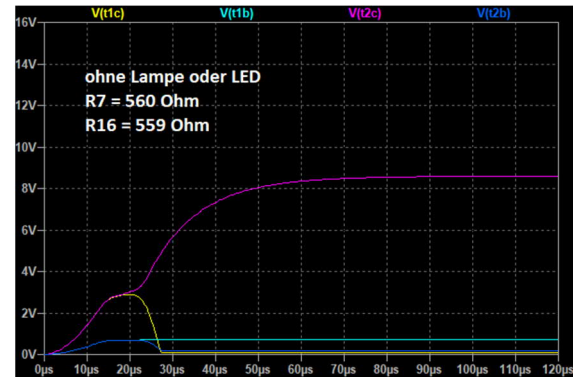


Abb. 25

In Abb. 25 sehen wir, was passiert, wenn R16 nur 1 Ω niedriger ist als R7. Wir sehen, dass der Transistor T1 geöffnet ist und die Spannung am Kollektor von T1 (gelb) auf „0“ zieht. Der Transistor T2 sperrt, wodurch die Spannung am Kollektor von T2 (violett) hoch wird.

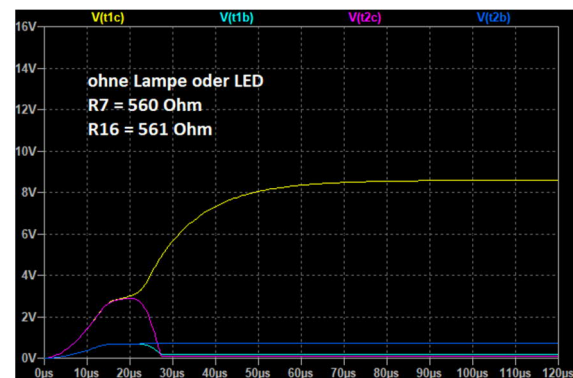


Abb. 26

Abb. 26 zeigt dasselbe, aber jetzt ist R16 nur 1 Ω höher als R7. Wir sehen jetzt, dass der Transistor T2 geöffnet ist und die Spannung am Kollektor von T2 (violett) auf „0“ zieht. Der Transistor T1 sperrt, wodurch die Spannung am Kollektor von T1 (gelb) hoch wird.

Fazit

Aufgrund eines EU-Gesetzes ist es nicht mehr erlaubt, Glühbirnen herzustellen und zu verkaufen. Wenn die Lampen ausfallen, müssen wir auf LEDs umsteigen. Allerdings unterscheiden sich die Eigenschaften einer LED wesentlich von denen einer Lampe. Bei unserem Flip-Flop können wir die Lampe also nicht einfach durch eine LED ersetzen.

Wollen wir weiterhin den Standard-Gleichrichterbaustein verwenden, müssen Anpassungen vorgenommen werden. Je nach verwendeter LED muss der Strom durch Erhöhen von R19 begrenzt werden. Der Wert von R16 muss jedoch verringert werden, um den viel höheren Widerstand der LED im Vergleich zur Lampe auszugleichen.

Werden die beschriebenen Änderungen vorgenommen, spielt es keine Rolle mehr, welche Spannungsversorgung ihr verwenden möchtet. Ich empfehle aber die Verwendung eines stabilisierten und einstellbaren Geräts.

Ich hoffe, dass ich mit diesem Beitrag mehr Klarheit darüber geschaffen habe, wie ein Flip-Flop funktioniert – mit Lampe oder mit LED.

Quellen

- [1] ftc-Forum: *Silberlinge*. Im [Forum der fischertechnik Community](#), 2021.
- [2] Analog Devices: *LTspice*. <https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html>
- [3] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 2)*. [ft:pedia 2/2021](#), S. 80–89.

Computing

Sensoren am TXT: Farberkennung mit der Kamera

Kurt Mexner

Mit verschiedenen Sensoren werden die Möglichkeiten des TXT Discovery Sets erheblich erweitert. Mit dieser Reihe möchte ich einige dieser Sensoren vorstellen. In diesem Beitrag beschäftigen wir uns mit der Farberkennung mittels der bereits mitgelieferten Kamera.

Die Kamera

In meinem letzten Beitrag [1] habe ich mit der Bewegungserkennung eine der Nutzungsmöglichkeiten der fischertechnik-USB-Kamera mit ROBO Pro vorgestellt. In diesem Beitrag geht es nun um die Erkennung von Farben.

Das Sensorfeld Farbe

Sobald man in ROBO Pro ein Programm öffnet oder neu erstellt, kann man den Reiter „Kamera“ anklicken. Ist eine USB-Kamera an den TXT (oder den PC) angeschlossen, wird das von der Kamera erfasste

Bild angezeigt. Hierfür muss im linken oberen Bereich das Häkchen „Kamera einschalten“ gesetzt sein (Abb. 1).

Im Menu ganz links oben findet sich unter „Elementgruppen“ die Eintragung „Sensorfelder“. Klickt man sie an, werden die verschiedenen Sensorfelder angezeigt. Wir klicken nun das Feld Farbe an und gehen mit der Maus auf den Kamerabildschirm. Der Mauszeiger verwandelt sich in einen Bleistift, mit dem wir jetzt ein Viereck beliebiger Größe in dem Kamerabild platzieren können. Mit einem Doppelklick wird es fertiggestellt.

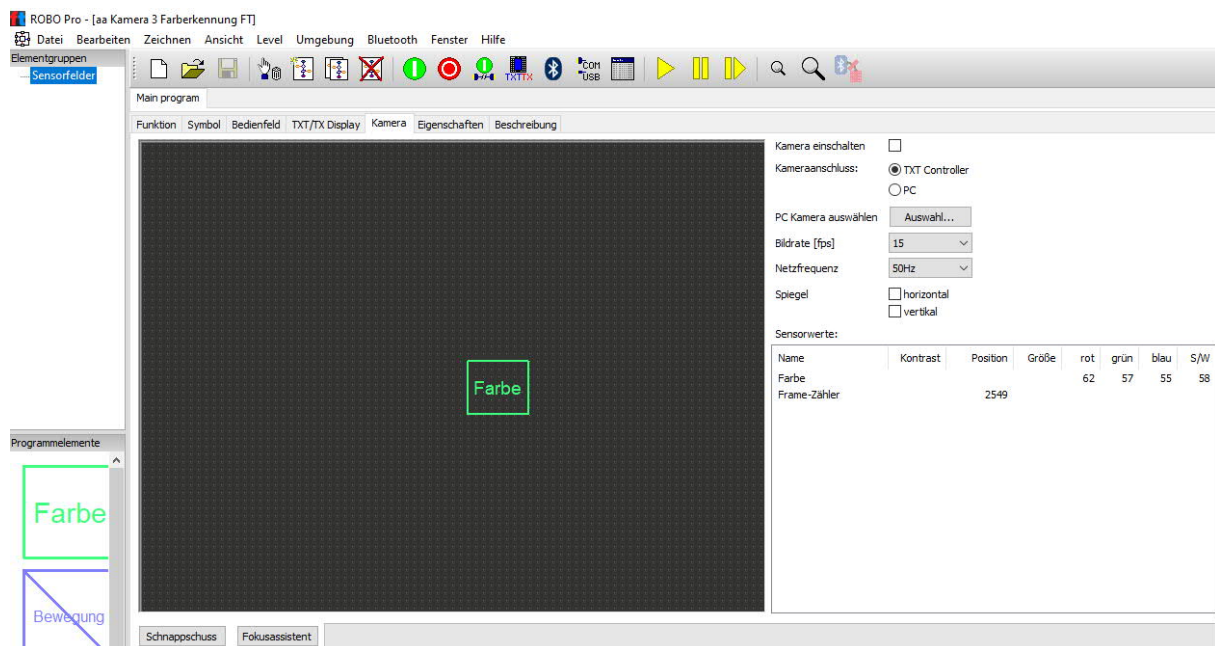


Abb. 1: Sensorfeld „Farbe“ in der Kamera-Konfiguration von ROBO Pro

Auch ohne Programm kann man nun testen, ob die Kamera eine Farbe erkennt. Dazu wird der grüne Start-Knopf gedrückt. Rechts im Kamera-Fenster werden nun in einer Tabelle für jedes Sensorfeld die Sensorwerte für „rot“ (R), „grün“ (G), „blau“ (B) und „S/W“ (schwarz/weiß) angezeigt. Alle anderen Farben muss man aus der Kombination dieser RGB-Werte ermittelt.

Wenn man nun z. B. rote Legosteine im Sensorbereich platziert, ändern sich die Sensorwerte und man erkennt, dass bei „rot“ der höchste Wert angegeben ist. Es ist möglich, mehrere Sensorfelder anzulegen. Jedem Feld muss dann ein anderer Name zugeteilt werden.

Das Programm

Im [Programm in Abb. 2](#) werden vier Unterprogramme aufgerufen. Jedes Programm ist für eine Farbe zuständig.

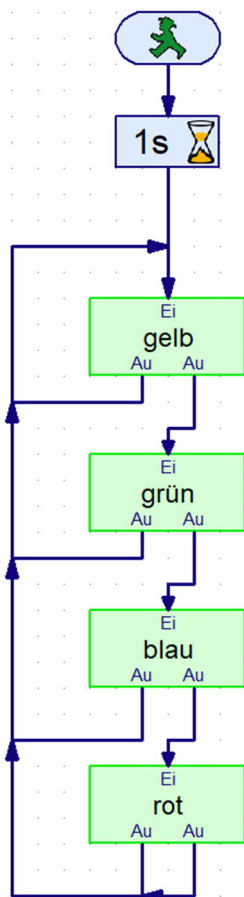


Abb. 2: ROBO Pro-Programm zur Farberkennung mit der USB-Kamera am TXT

Betrachten wir nun das Unterprogramm, das für Rot zuständig ist (Abb. 3). Der Wert für „rot“ wird mit dem Wert für „grün“ verglichen. Ist „rot“ größer, wird eine 1 ausgegeben, ansonsten 0. Im nächsten Schritt wird „rot“ mit „blau“ verglichen. Auch hier gibt es eine 1, wenn „rot“ größer ist.

Vor allem bei Kunstlicht werden manchmal falsche Ergebnisse geliefert. Deshalb ist als zusätzliche Abfrage ein Vergleich mit dem Sensorwert b/w (black/white) notwendig. Zu diesem Wert wird 6 dazu addiert. Wenn dieser neue Wert kleiner ist als „rot“, wird ebenfalls eine 1 ausgegeben.

Anschließend vergleicht der Operator „=“ („ist gleich“) alle drei Werte, ob sie mit 1 übereinstimmen. Wenn dies zutrifft erhält das Warten-auf-Element eine 1 und die Anzeigelampe auf dem Monitor leuchtet rot auf.

Anlog laufen die Unterprogramme „blau“ und „grün“ ab.

Alle anderen Farben werden aus der Kombination der RGB-Werte ermittelt. Das kann aufwändig sein: Die Farbe Gelb z. B. hat mich mehrere Tage Arbeit gekostet. Die Bedingungen für Gelb lauten:

- die Differenz von „rot“ und „grün“ ist maximal 17
- der Wert für „blau“ liegt zwischen -2 und 32

Bei starker Sonneneinstrahlung liegen alle vier Werte des Sensorelementes sehr nahe beieinander. Eine Differenzierung ist dann oft ausgeschlossen. In diesem Extremfall sollte das Sensorfeld abgeschattet werden.

Ich testete das Programm mit normalem Tageslicht, Glühlampenlicht und mit einer LED-Taschenlampe. Alle vier Farben wurden korrekt erkannt.

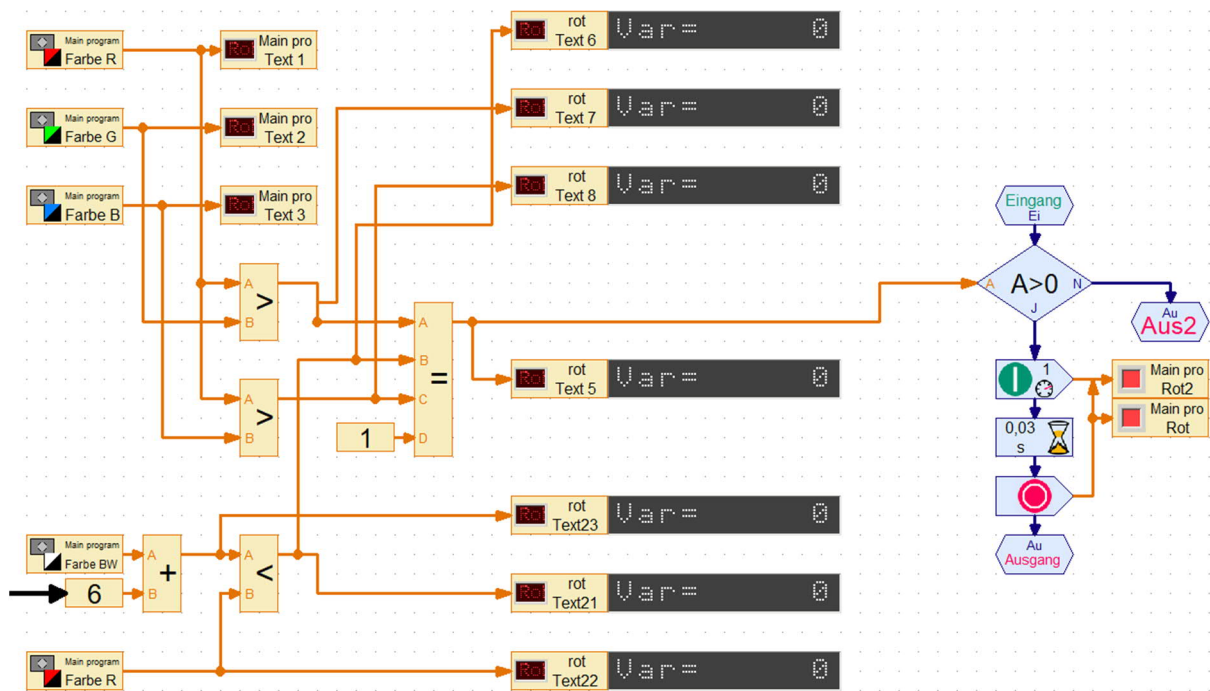


Abb. 3: Unterprogramm „rot“ zur Erkennung der Farbe Rot

Es ist auch möglich die Kamera mit normalen farbigen LED anzustrahlen (oder auch einer RGB-LED). Dies funktioniert aber nur mit roten, grünen und blauen LED. Eine gelbe LED funktioniert nicht. Auch eine IR-LED wird von der Kamera nicht erkannt.

Einstellungen

Alle Werte, die im ROBO Pro-Programm mit einem Pfeil markiert sind, dienen zur Anpassung an bestehende Lichtverhältnisse, Farbintensitäten oder Farbnuancen. Ich habe die Werte an die LEGO-Farben angepasst, da ich davon ausgehe, dass diese Farben bei jedem verfügbar sind.

Tageslicht, Kunstlicht oder LED-Licht liefern je nach Situation unterschiedliche Ergebnisse, die solche Anpassungen notwendig machen können.

Bedienung

Ich richte die Kamera auf ein weißes Blatt Papier und schiebe in das Sensorfeld die zu erkennende Farbe (Legostein, farbiges Papier, sonstiges Objekt). Die Kamera ist auf einem fischertechnik-„Stativ“ befestigt und zeigt nach unten (Abb. 4).

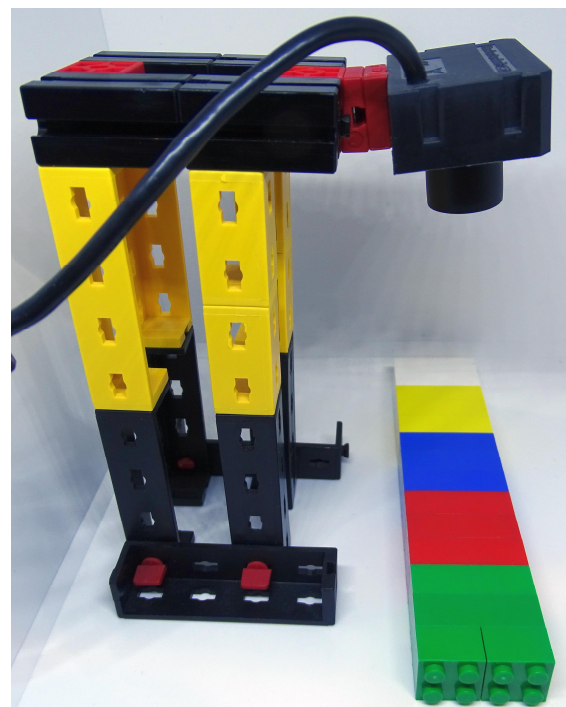


Abb. 4: Kamera-„Stativ“

Im ROBO Pro-Programm werden im oberen Bereich in den grau unterlegten Feldern die Sensorwerte der Kamera für „B/W“, „rot“, „grün“ und „blau“ angezeigt. Daneben leuchtet ein Lampenelement auf, wenn eine Farbe erkannt wird (Abb. 5).

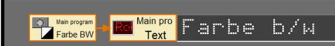
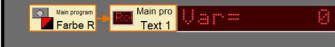

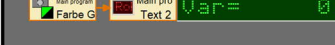

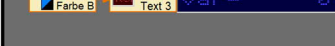


Ausgelesene Werte des Kamera-Sensor-Feldes	Erkannte Farbe
	
	
	
	
	

Abb. 5: Anzeige der Farberkennung

Anwendungen

Die folgenden Anwendungen der Farberkennung in einem Modell sind vorstellbar:

- auf einem Fließband werden die farbigen Steine erkannt und sortiert.
- ein Greifarm sortiert die Steine
- Programmsteuerung über farbige Karten

- Fernsteuerung über farbige LED

Die Farb-Erkennung funktioniert auch über weite Entfernungen

Nachtrag

Aktuell gibt es den Baukasten Robotics Smartech sehr günstig zu kaufen (für ca. 180 €). Er enthält den Robotics TXT Controller, die Software ROBO Pro, vier Minimotoren mit Getriebe, einen Gestensensor, einen Spursensor, vier Mecanum Omniwheels (faszinierende Räder, die sich seitwärts bewegen können) und ein Hubgetriebe. Zu diesem Preis ist dies ein Schnäppchen, mit dem man preisgünstig in die Computerprogrammierung einsteigen kann. Nicht vergessen, ein Netzteil oder einen Akku mitzubestellen.

Referenzen

- [1] Kurt Mexner: *Sensoren am TXT: Die Kamera*. [ft:pedia 1/2022](#), S. 112–116.

Computing

Einführung in ftScratch (3): Der Barcodeleser

Dirk Fox

Scratch ist eine für Einsteiger und Schulen besonders geeignete Programmiersprache. Die Entwicklungsumgebung ftScratch3 unterstützt mit entsprechenden Erweiterungen die fischertechnik-Controller TXT und BT Smart [1]. In dieser Serie führen wir in die Programmierung mit ftScratch ein – mit kleinen Modellen und Aufgaben.

Barcodes

Ein Barcode ist ein Balken- oder Strichcode, der sich aus schwarzen und weißen, unterschiedlich breiten Balken zusammensetzt. Barcodes zählen zu den eindimensionalen Binärcodes, da die Information allein in der Breite der Balken enthalten ist und nur die Werte „0“ und „1“ kodiert werden (bspw. schmaler Balken = „0“, breiter Balken = „1“). Die Farbe der Balken (schwarz, weiß) dient zur optischen Unterscheidung benachbarter Balken.

Barcodes sind optische Codes: Sie können von einem Sensor, der „hell“ und „dunkel“ unterscheiden kann, oder von einer Kamera mit Bildauswertung gelesen und dekodiert werden. Barcodes werden überwiegend zur Kennzeichnung von Objekten verwendet; man findet sie z. B. auf Büchern und Zeitschriften (ISBN/ISSN), auf Waren-Etiketten oder Aufklebern für die Paketpost.

Es gibt sehr viele unterschiedliche Barcodes, die sich durch ihre Informationsdichte und durch die verwendeten Prüfwerte unterscheiden [2]. Barcodes haben den Vorteil, dass sie sehr robust sind, also auch verschmutzt oder beschädigt meist noch lesbar sind. Sie lassen sich sehr leicht einlesen, und die Lesegeräte sind vergleichsweise günstig und weit verbreitet – in Kassensystemen sind sie heute ein Standard.

Der „Code 39“

Ein verbreiteter Barcode ist der Code 39 (Abb. 1, siehe auch [3]). Er wurde 1973 bei der Fa. Intermec entwickelt und seitdem von diversen Normungs-Organisationen (darunter ANSI und ISO) standardisiert, u. a. 1999 als ISO/IEC 16388 [4].

Ein Code-39-Zeichen besteht aus neun Balken: fünf schwarzen und vier weißen (den Zwischenräumen) in zwei verschiedenen Breiten. Vor und hinter jedem Zeichen folgt ein (schmaler) weißer Zwischenraum, damit man die einzelnen Zeichen unterscheiden kann. Der Code-39-Standard definiert insgesamt 44 Zeichen: die 26 Buchstaben des Alphabets, die zehn Ziffern „0“ bis „9“ (Abb. 1) und acht Sonderzeichen – Leerzeichen, „-“, „+“, „.“, „/“, „%“, „\$“ und „*“ (Start-/Stoppzeichen). In Tabelle 1 am Ende des Beitrags findest du alle Zeichen des Code-39 in einer Übersicht.

Code-39-Barcodes sind immer gleich breit, denn die Anzahl der breiten Balken je Zeichen ist auf genau drei festgelegt. Bei Buchstaben und Ziffern sind es immer zwei schwarze und ein weißer.

Ein Code-39-Zeichen besteht also immer aus sechs schmalen und drei breiten Balken. Wenn wir einen breiten Balken als „1“ und einen schmalen als „0“ interpretieren, repräsentiert ein Code-39-Zeichen eine neunstellige Binärzahl.



Abb. 1: Code-39-Alphabet

Der Code-39 entspricht damit einem binären „3-aus-9“-Code. Damit bezeichnet man einen neunstelligen Binärcode, dessen Codeworte jeweils genau drei Einsen enthalten. Mit einem 3-aus-9-Code können

$$\binom{9}{3} = \frac{9!}{3! \cdot (9-3)!} = \frac{9 \cdot 8 \cdot 7 \cdot 6!}{3 \cdot 2 \cdot 6!} = 84$$

unterschiedliche Zeichen kodiert werden.

Tatsächlich umfasst der Code-39 lediglich 44 Zeichen; nur etwa jedes zweite Zeichen des 3-aus-9-Codes ist ein gültiges Code-39-Zeichen. Damit liegt der Informationsgehalt des Code-39 bei sechs Bit ($2^6 = 64$).

Mit neun Bit sind $2^9 = 512$ verschiedene Zahlen darstellbar. Damit entspricht nur etwa jede zwölfte neunstellige Binärzahl einem gültigen Code-39-Zeichen. Der Code-39 hat also eine sehr hohe *Redundanz*: Er ist nicht sehr effizient (sechs Bit würden für die Kodierung von 44 Zeichen ausreichen); dafür werden Lesefehler mit einer

sehr hohen Wahrscheinlichkeit erkannt. Da je zwei Zeichen des Code-39 sich an mindestens zwei Stellen unterscheiden (Hamming-Abstand 2), sind sogar Fehlerkorrekturen möglich. Aber Achtung: Der Code ist nicht gegen „Umkehrung“ geschützt: wird der Code falsch herum eingelesen, liefert die Dekodierung ein anderes Zeichen (bspw. „1“ statt „A“, „2“ statt „H“ usw.).

Der Barcodeleser

Nun möchten wir Code-39-Barcodes mit Scratch und unserem fischertechnik-Controller einlesen und dekodieren. Das machen wir mit einem kleinen mechanischen Modell, das den Code „abtastet“: Wir lassen eine an einer Laufschiene angebrachte, von einem Motor angetriebene Leseinheit aus einer Linsenstecklampe (37875) oder einer Lichtschranken-LED (162135) und einem Fototransistor über den Code „hinwegfahren“ (Abb. 2).

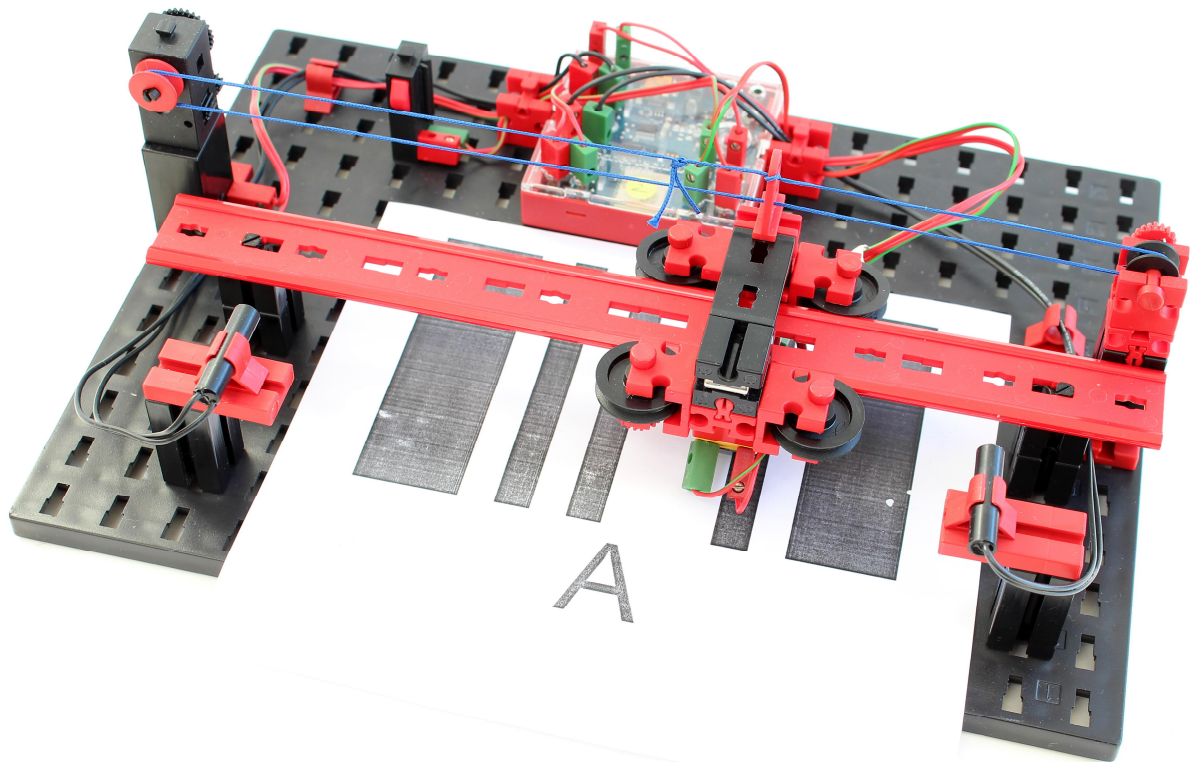


Abb. 2: Barcode-Leser

Das geht nicht so schnell wie mit einem CCD-Sensor oder einer Kamera; dafür lernt man dabei, wie das Erkennungsverfahren im Detail funktioniert.

Baue zunächst den Barcodeleser aus Abb. 2 auf. Die Designer-Datei des Modells findest du im [Downloadbereich dieser Ausgabe](#) der ft:pedia.

Ein paar Hinweise zum Aufbau des Modells:

- Das fischertechnik-Seil muss du mindestens einmal um die kleine Führungsrolle wickeln, damit es greift.
- Die Endlagenerkennung erfolgt mit einem kleinen zylindrischen Neodym-Magneten in der unteren Nut des BS15 auf dem Lesekopf. Zu kaufen gibt es solche Stabmagnete mit 4 mm Durchmesser z. B. bei [Supermagnete.de](#). Alternativ kannst du einen flachen Neodym-Magneten, wie er sich häufig in „Edelverpackungen“ findet, auf eine Bauplatte 15·15 kleben und diese in der unteren Nut des BS15 befestigen.

- Justiere die Fotodiode und die Linsenlampe am Lesekopf so, dass in der Anzeige im Gateway deines Controllers bei weißem Untergrund der Wert „1“ und bei dunklem „0“ an I1 angezeigt wird.

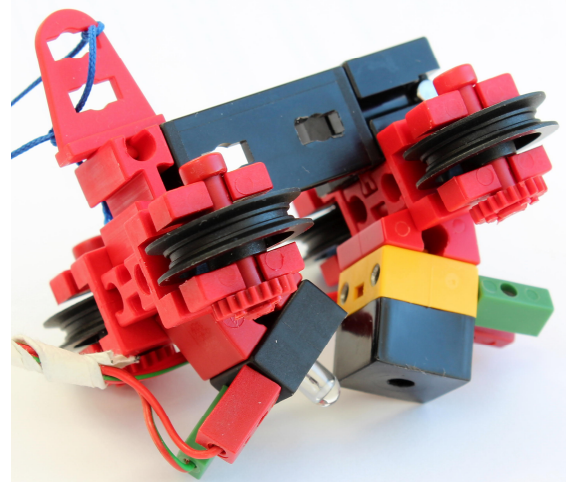


Abb. 3: Lesekopf (von unten)

- Die Position der Reed-Kontakte musst du ebenfalls ein wenig justieren: Der Reed-Kontakt muss „anschlagen“, bevor

Diode und Lampe unterhalb des Schlittens die BS15 berühren, die die Laufschiene tragen.

Sensoren und Aktoren

Die Aufgaben können sowohl mit dem TXT als auch mit dem BT Smart Controller gelöst werden. Für die Aufgaben benötigst du

- eine Fotodiode,
- eine Linsenstecklampe oder eine Lichtschranken-LED,
- einen XS-Motor,
- einen Taster und
- zwei Reed-Kontakte.

Die Sensoren und Aktoren werden wie folgt angeschlossen:

- I1: Fotodiode
- I2: Reed-Kontakt an der Endlage „Startposition“
- I3: Reed-Kontakt an der Endlage „Endposition“
- I4: Start-Taster
- M1: XS-Motor
- M2: Linsenstecklampe/Lichtschranken-LED

Aufgaben

Für die folgenden Aufgaben kannst du die Barcodes aus dem Code-39-Alphabet im [Downloadbereich dieser Ausgabe](#) der ft:pedia verwenden. Drucke die gewünschten Zeichen aus und schneide auf beiden Seiten einen etwa 1,5 cm breiten Streifen ab, damit der Code unter den Lesekopf passt. Wir beginnen zunächst mit der Steuerung des Lese-Kopfes, dann „messen“ wir unseren Barcode aus und schließlich vervollständigen wir den Leser um die Dekodierung des eingelesenen Codes.

Aufgabe 1

Der Lese-Kopf des Barcode-Lesers soll zu Beginn in die Startposition fahren. Wird der Start-Taster (I4) betätigt, soll sich der Lese-Kopf (mit moderater Geschwindigkeit) in Bewegung setzen, bis er die Endlage erreicht hat und dort stoppen. Wird der Start-Taster erneut gedrückt, soll der Lesekopf zurück in die Startposition fahren.

Aufgabe 2

Jetzt legen wir den Testbogen mit dem Barcode in unseren Barcodeleser ein. Während der Lese-Kopf sich in Richtung Endlage bewegt, wollen wir die Breite der von der Fotodiode erkannten Balken des Barcodes messen, indem wir die Zeit stoppen, die bis zum Beginn des nächsten Balkens verstreicht.

Die Messwerte sollen in eine Liste eingetragen werden, damit wir daraus einen Schwellenwert bestimmen können, anhand dessen sich eine „0“ (schmaler Balken) von einer „1“ (breiter Balken) unterscheiden lässt.

Miss‘ die Balkenbreiten deines Barcodes bei verschiedenen Motorgeschwindigkeiten. Du siehst, dass die Unterscheidung bei niedrigen Geschwindigkeiten einfacher ist. Lege einen geeigneten Schwellenwert für eine passende Geschwindigkeit des Motors fest.

Tipp: Arbeite mit Threads.

Aufgabe 3

Nun können wir den ersten Barcode dekodieren. Lege den Code-39 des Buchstabens „A“ oder „B“ unter deinen Barcodeleser. Werte die Balken nun als Binärzahl aus: Ein schmaler Balken entspricht einer „0“ an der entsprechenden Binärstelle, ein breiter einer „1“. Erweitere dein Programm aus Aufgabe 2 um die Bestimmung des Binärwertes des Codes und zeige das Ergebnis im Fenster der Bühne an.

Hinweis: Aus der Tabelle am Ende des Beitrags kannst du entnehmen, dass der Code für „A“ den Wert 265, der für „B“ den Wert 73 liefern muss.

Aufgabe 4

Jetzt fehlt noch die Dekodierung des Binärwerts: Der Barcodeleser soll nicht den Binärcode, sondern das kodierte alphanumerische Zeichen anzeigen. Dazu benötigst du zwei Listenvariablen: Eine, die alle gültigen Binärwerte des Code-39 enthält, und eine mit allen zugehörigen alphanumerischen Zeichen (Tabelle 1). Um dir das Abtippen zu ersparen gibt es beide Listen im [Downloadbereich dieser Ausgabe](#) der ft:pedia. Importiere die Daten in je eine Listenvariable und erweitere dein Programm aus Aufgabe 4 so, dass statt des Binärcodes das kodierte alphanumerische Zeichen auf der Bühne angezeigt wird.

Aufgabe 5

Erweitere dein Programm aus Aufgabe 4 so, dass nacheinander mehrere Barcodes eingelesen werden können und die Dekodierung als Zeichenfolge angezeigt wird.

Aufgabe 6

Ergänze dein Scratch-Skript nun noch um eine Sprachausgabe, die die eingelesene Buchstabenfolge vorliest.

Lösungsbeispiele

Die folgenden Lösungsbeispiele wurden für den BT Smart Controller programmiert. Die entsprechenden Scratch-Skripte und alle Hilfsdateien findet ihr im [Downloadbereich dieser Ausgabe](#) der ft:pedia. Die Anpassung der Skripte an den TXT Controller (Ersetzung der entsprechenden Blöcke in ftScratch) ist sehr einfach.

Aufgabe 1

Die Bewegung in die Startposition wurde in einen eigenen Block ausgelagert, damit das Programm übersichtlicher wird.



Abb. 4: Bewegung des Lesekopfes

Aufgabe 2

Die Messung gelingt einfach, wenn wir in zwei parallelen Threads bei jedem Signalwechsel der Fotodiode („Schließen“ oder „Öffnen“) den Wert der Stoppuhr speichern und die Stoppuhr zurücksetzen (Abb. 5).

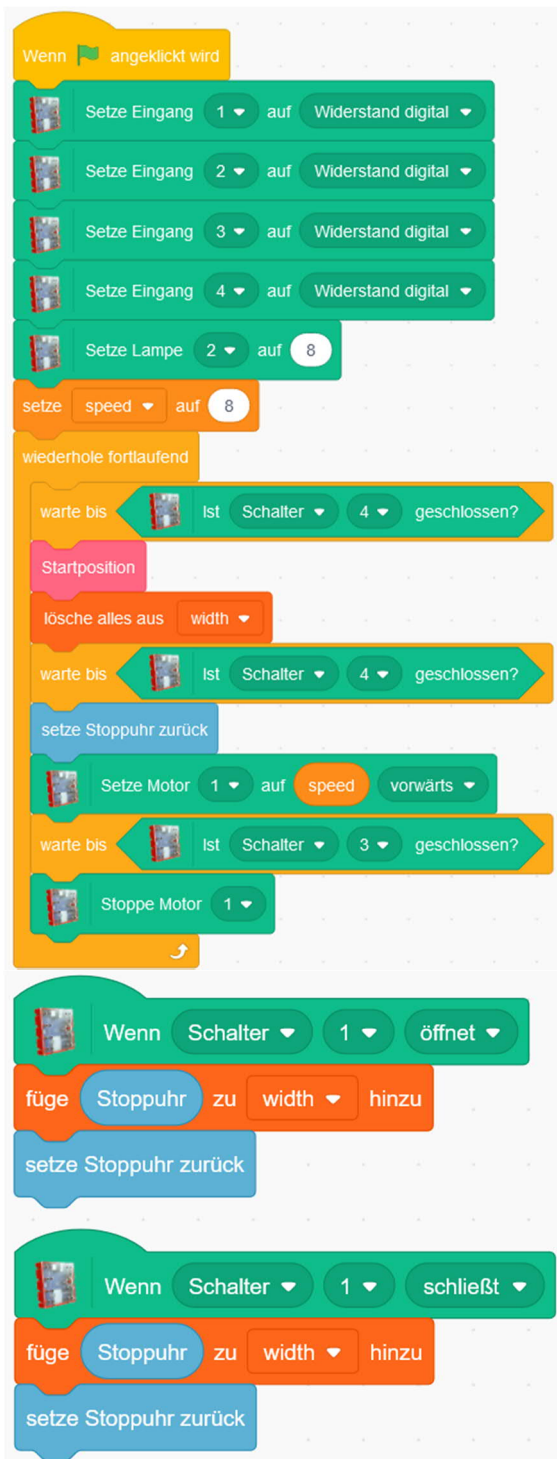


Abb. 5: Messung der Balkenbreiten

Der erste Eintrag in der Liste ist die Zeit bis zum Beginn des ersten Balkens, daher werden insgesamt zehn Messungen vorgenommen. Im Testmodell konnten die Balken auch bei Maximalgeschwindigkeit (8) noch gut unterschieden werden: schmale Balken wurden vom Lesekopf in weniger als 150 ms, breite Balken in über 350 ms überfahren. Als Schwellenwert werden daher 250 ms festgelegt.

Aufgabe 3

Die Bestimmung des Binärwerts des Codes gelingt durch eine einfache Erweiterung der beiden Threads, die auf das Schließen bzw. Öffnen der Fotodiode reagieren (Abb. 6). Dabei wird der Balkencode am einfachsten von links (höchstwertiges Bit) nach rechts ausgewertet:

- Setze die Ergebnisvariable auf 0
- Multipliziere die Ergebnisvariable bei jedem Wechsel des Input-Signals mit zwei.
- Wenn die gemessene Zeit seit dem letzten Wechsel größer als der Schwellenwert ist, addiere zur Ergebnisvariable 1.



Abb. 6: Bestimmung der Stellen des Binärwerts

Da das höchstwertige Bit der eingelesenen Zahl durch den Freiraum vor dem ersten Balken zustande kommt, musst du es zum Schluss löschen. Das geht am einfachsten, indem du den Rest des Werts der Ergebnisvariable modulo $2^9 = 512$ bestimmst.

Aufgabe 4

Mit den beiden Listenvariablen ist die abschließende Dekodierung des Code-39-Zeichens geradezu trivial: Es genügt eine einzige Programmzeile, um aus dem Binärwert das kodierte Zeichen zu erhalten (Abb. 7).



Abb. 7: Umwandlung des Binärwerts in das dekodierte alphanumerische Zeichen

Die kurze Wartezeit stellt sicher, dass die Modulo-Berechnung erst nach der Auswertung des letzten Signal-Wechsels an I1 erfolgt. Den Wert der Variable „character“ kann man sich auf der Bühne anzeigen lassen.

Aufgabe 5

Statt „character“ verwenden wir die Variable „message“ und hängen jedes neu dekodierte Zeichen einfach an (Abb. 8).



Abb. 8: Anhängen der dekodierten Zeichen

Aufgabe 6

Die Sprachausgabe ist leicht ergänzt: Zu Beginn des Skripts werden Sprache und Stimmfarbe gewählt, dann erfolgt zum Schluss die Sprachausgabe (Abb. 9).



Abb. 9: Sprachausgabe

Weiterentwicklung des Modells

Die Geschwindigkeit des Barcodelesers kannst du verdoppeln, wenn bei jeder Bewegung des Lesekopfes eine Auswertung des eingelegten Barcodes erfolgt. Entweder legst du dazu jeden zweiten Barcode auf dem Kopf stehend ein, oder du liest den Code abwechselnd beginnend beim höchst- resp. niederstwertigen Bit ein. Letzteres macht den Lesevorgang etwas komplexer.

Mit dem zweiten Motorausgang kannst du auch einen Transportmotor ansteuern. Damit kannst du von einem Stapel gedruckter Code-39-Zeichen eines nach dem anderen in Folge einlesen, ohne dass du die Zeichen manuell einlegen musst. Den Start-Taster kannst du in dieser Modellvariante durch eine Lichtschranke ersetzen, die den Motor stoppt, wenn das nächste Zeichen in der richtigen Position liegt.

Du kannst das Modell auch so umbauen, dass nicht mehr einzelne Code-Zeichen, sondern ein langer Streifen mit einem aufgedruckten Code-39-Barcode gelesen werden kann. Der Lesekopf kann dann fest montiert werden, und der Barcode wird mit einem Motor in gleichmäßiger Geschwindigkeit darunter vorbeigeschoben.

Um einen zusammenhängenden Barcode zu erzeugen, kannst du dir entweder eine Code-39-Schriftart für dein Textverarbeitungsprogramm installieren (z. B. [Code-39-Logitogo](#)) oder dir die Zeichen über einen kostenlosen Online-Dienst generieren lassen (z. B. von [tec-it](#)). Dabei musst du beachten, dass ein vollständiger Barcode immer vor dem ersten und hinter dem letzten Zeichen den Start-Stopp-Code (*) enthält. Daran kannst du leicht den Anfang und das Ende eines Barcode-Textes erkennen.

Code-39 (binär)	dezimal	Zeichen
000110100	52	„0“
100100001	289	„1“
001100001	97	„2“
101100000	352	„3“
000110001	49	„4“
100110000	304	„5“
001110000	112	„6“
000100101	37	„7“
100100100	292	„8“
001100100	100	„9“
100001001	265	„A“
001001001	73	„B“
101001000	328	„C“
000011001	25	„D“
100011000	280	„E“
001011000	88	„F“
000001101	13	„G“
100001100	268	„H“
001001100	76	„I“
000011100	28	„J“
100000011	259	„K“
001000011	67	„L“
101000010	322	„M“
000010011	19	„N“
100010010	274	„O“
001010010	82	„P“
000000111	7	„Q“
100000110	262	„R“

Code-39 (binär)	dezimal	Zeichen
001000110	70	„S“
000010110	22	„T“
110000001	385	„U“
011000001	193	„V“
111000000	448	„W“
010010001	145	„X“
110010000	400	„Y“
011010000	208	„Z“
010000101	133	„-“
110000100	388	„.“
011000100	196	„ “
010101000	168	„\$“
010100010	162	„/“
010001010	138	„+“
000101010	42	„%“
010010100	148	„*“

Tab. 1: Code-39

Referenzen

- [1] Dirk Fox: *Scratch mit fischertechnik – Update 2022*. [ft:pedia 1/2022](#), S. 86–92.
- [2] Wikipedia: [Strichcode](#).
- [3] Andreas Gail: *Strichcode-Leser am Robo TX Controller (1): Automatisiert mit RoboPro*. [ft:pedia 3/2014](#), S. 66–71.
- [4] International Organization for Standardization (ISO): *ISO/IEC 16388:2007. Information technology — Automatic identification and data capture techniques — Code 39 bar code symbology specification*, 15.05.2007, confirmed 2014.

Computing

Android App für den BT-Empfänger mit dem MIT App Inventor 2

Thomas Kaiser

Der Bluetooth Empfänger kann auch mit einer von fischertechnik bereitgestellten Smartphone-App gesteuert werden, die durchaus ihren Zweck erfüllt. Aber was, wenn man eigene Ideen hat?

Einführung

Eine App für ein Android Smartphone selbst programmieren – dazu muss man nicht unbedingt Java oder Kotlin beherrschen. Vor einiger Zeit fand ich auf der Suche nach Alternativen den „[MIT App Inventor](#)“, der es ermöglicht, für Android (und seit kurzem auch Apple) Smartphones Apps grafisch zu programmieren.

Ursprünglich 2010 von Google entwickelt, wird der MIT App Inventor seit 2012 vom MIT (Massachusetts Institute of Technology) [weiterentwickelt](#).

Grund für meine Suche war ursprünglich der Wunsch, eine App zum Steuern eines Arduino per Bluetooth zu entwickeln. Und es stellte sich heraus, dass es ebenfalls möglich ist, den fischertechnik-BT-Empfänger durch eine mit dem MIT App Inventor 2 erstellte App anzusteuern.

Die vom fischertechnik-BT-Empfänger genutzte Technik, es handelt sich um ein [Bluetooth Low-Energy](#) (BLE) Gerät, soll hier nur kurz angerissen werden, da eine umfassende Erklärung den Rahmen des Beitrags sprengen würde. Der BT-Empfänger nutzt das GATT-Profil, um sich mit dem Smartphone zu verbinden. Dabei teilt das Gerät selbst mit, welche Dienste (Services) und Eigenschaften (Characteristics) angeboten werden.

Angesprochen werden die entsprechenden Eigenschaften dann über sog. [UUIDs](#) (siehe hierzu auch den Anhang „Wie man die UUIDs herausfindet“).

Die UUIDs könnten von der App auch aus dem Empfänger ausgelesen werden; das führt aber schnell zu unübersichtlichen Programmen. Um die Programme einfacher zu halten, werden die UUIDs in diesen Beispielen einfach als Festwert definiert.

Hinweis: Zum gegenwärtigen Zeitpunkt (2022) ist der App-Inventor für das iPhone noch in der Entwicklung – nach meinen bisherigen Recherchen scheint BLE noch nicht unterstützt zu werden.

Voraussetzungen

Um Programme mit dem App Inventor 2 zu erstellen und ausführen zu lassen, braucht man auf dem PC/Laptop einen Browser (getestet habe ich mit Firefox und Edge) und auf dem Smartphone (oder auch Tablet) die App „MIT AI2 Companion“ (für Android zu finden im Play Store).

Im Idealfall ist das Smartphone im gleichen WLAN eingeloggt wie der PC; in diesem Fall benötigt man nicht einmal eine USB-Verbindung.

Zuerst müssen wir uns beim MIT App-Inventor einloggen. Sofern man einen Google-Account hat, kann man diesen verwenden; alternativ kann man sich auch ohne Account anonym einloggen – in diesem Fall

sollte man sich unbedingt den „Revisit Code“ notieren, um weiterhin auf seine in der Cloud gespeicherten Programme zugreifen zu können. Eine Login-Seite, die beide Möglichkeiten anbietet, ist erreichbar unter <http://code.appinventor.mit.edu>.

Nachdem man sich eingeloggt hat, muss man einige Hinweisfenster bestätigen, um schließlich zum Hauptfenster zu kommen.

Rechts oben in der Menü-Leiste lässt sich die Sprache auf „Deutsch“ ändern (Abb. 1).¹ Je nach Einstellung des Browsers muss man u. U. ein Hinweisfenster bestätigen, das Pop-Up-Fenster erlaubt.

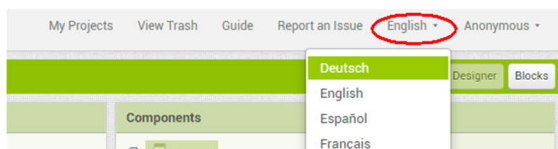


Abb. 1: Einstellung der Sprache

Erstellen einer Android App

Im Folgenden werden wir eine einfache App erstellen und Schritt für Schritt erweitern, die den fischertechnik-BT Empfänger ansteuern kann. Der fertige Code kann auch direkt von meinem [Github-Repository](#) heruntergeladen werden (Datei „FTBT_1_4Schieberegler.aia“).

Die Programme des App-Inventor sind Ereignis-orientiert. Anders als bei ROBO Pro, bei dem man in einer Schleife Eingänge abfragt, wird hier automatisch beim Auslösen eines Ereignisses ein entsprechendes Unterprogramm ausgeführt. So wird zum Beispiel beim Drücken einer Taste das zugehörige Ereignis „wenn Taste.Klick mache“ ausgeführt. An dieser Stelle möchte ich auch auf die vom MIT zur Verfügung gestellten [Tutorials](#) hinweisen; man findet auch [viele Tutorials auf Deutsch](#).

Erster Schritt: Design erstellen

Mit der Schaltfläche „Neues Projekt starten“ legt man ein neues Projekt an (Abb. 2), das ich „FTBT_1“ genannt habe. Nun sollte sich der Design-Bereich des App-Inventors öffnen.

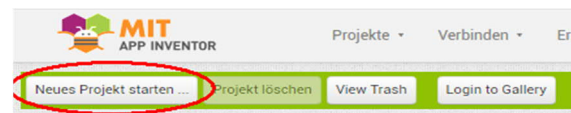


Abb. 2: Neues Projekt anlegen

Mit der linken Maustaste (gedrückt halten) können wir nun unsere erste Taste auf den Bildschirm ziehen (Abb. 3). Die Namen werden automatisch erzeugt; zur besseren Lesbarkeit werden wir sie im nächsten Schritt ändern.

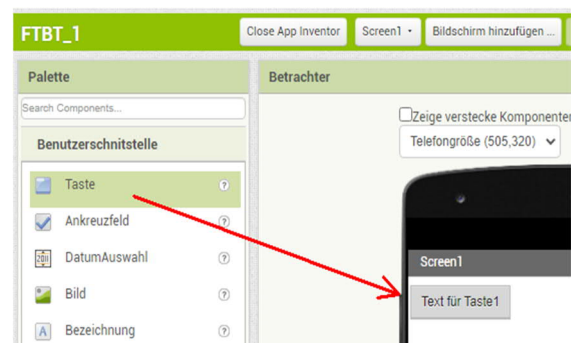


Abb. 3: Taste platzieren

Hierzu wählen wir bei „Komponenten“ die gerade erzeugte Taste durch Anklicken aus, klicken unten auf „Umbenennen“ und ändern den Namen auf „TasteSuchen“. Unter diesem Namen werden wir die Taste später in der Programmierumgebung wiederfinden, deshalb ist es sinnvoll, die Funktion der Taste im Namen anzugeben (Abb. 4).



Abb. 4: Name der Taste vorher ... und nachher

¹ Die deutsche Übersetzung des App Inventors ist leider nicht überall vollständig.

Im nächsten Schritt ändern wir den Text der Taste. Hierzu wählen wir rechts in der Spalte „Eigenschaften“ das Feld „Text“ (falls es nicht sichtbar ist, nach unten scrollen). Hier ändern wir den Text „Text für Taste1“ auf „Suchen“ (Abb. 5).

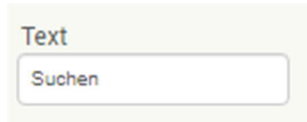


Abb. 5: Umbenennung der Text-Eigenschaft

Diese Änderung wird uns auch sofort im Betrachter-Fenster angezeigt. Nun brauchen wir noch eine Möglichkeit, Statusmeldungen oder Fehlermeldungen auszugeben. Dies wird im App-Inventor mit einer „Bezeichnung“ gelöst. Genauso wie vorher die Taste, ziehen wir die Bezeichnung auf den Smartphone-Bildschirm. Anschließend benennen wir die Bezeichnung um auf „Status“; den Text in den Eigenschaften können wir unverändert lassen.

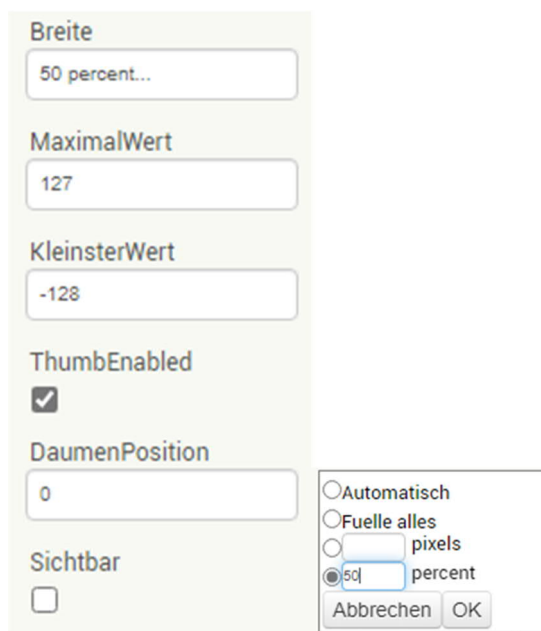


Abb. 6: Eigenschaften des Schiebereglers

In diesem ersten Versuch werden wir nur einen Motor (M1) ansteuern. Hierzu ziehen

wir aus der Palette einen Schieberegler auf den Bildschirm unter den Statustext. Den automatisch vergebenen Namen behalten wir bei, aber bei den Eigenschaften nehmen wir die in Abb. 6 gezeigten Änderungen vor.

Beim Anklicken der Breite öffnet sich ein Dialog, in dem man die Prozent-Angabe wählen und den Wert eintragen kann (Abb. 6, rechts). Den Haken bei „Sichtbar“ entfernen wir – dadurch verschwindet der Schieberegler aus der Ansicht. Wir können den Schieberegler später programmgesteuert wieder sichtbar machen.

Die Suche nach den Bluetooth-Geräten erstellt eine Liste der verfügbaren Geräte. Um diese Liste anzuzeigen und das gewünschte Gerät auszuwählen, nutzen wir eine „Listenansicht“. Diese Listenansicht ziehen wir aus der Palette und benennen sie in „ListeBT“ um. In den Eigenschaften stellen wir bei Breite „Fuelle alles“ und bei Höhe „50 percent“ ein (Abb. 7).



Abb. 7: Eigenschaften von „ListeBT“

Da der App Inventor von Haus aus zwar Bluetooth, aber nicht BLE ansteuern kann, müssen wir uns eine passende Extension (Erweiterung) herunterladen.² Hierzu geben wir im Browser den folgenden Link ein: <https://mit-cml.github.io/extensions/>.

In der Liste der unterstützten Erweiterungen steht an erster Stelle die Extension „BluetoothLE“. Mit einem Klick auf [BluetoothLE.aix](#) wird die Extension heruntergeladen

² Beispielanwendung der BLE-Extension: <https://iot.appinventor.mit.edu/assets/tutorials/MIT App Inventor Basic Connection.pdf>.

(üblicherweise in den Ordner „Downloads“ auf dem Rechner).

Für den Import klicken wir in der Palette links ganz unten auf „Extension“ und dann auf „Import extension“. Im nun erscheinenden Dialog klicken wir auf „Datei auswählen“ und wählen die gerade heruntergeladene .aix Datei aus. Mit einem Klick auf „Import“ wird die Datei installiert (Abb. 8).

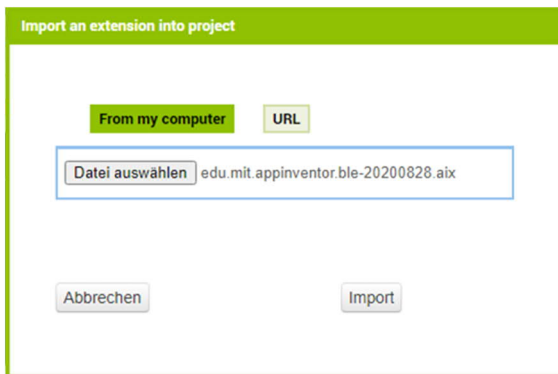


Abb. 8: Import der BLE-Erweiterung

Nun können wir links in der Palette die Erweiterung auswählen und ziehen sie herüber auf den Bildschirm. Die Erweiterung wird automatisch unten im Bereich „nicht sichtbare Komponenten“ eingeordnet (Abb. 9).

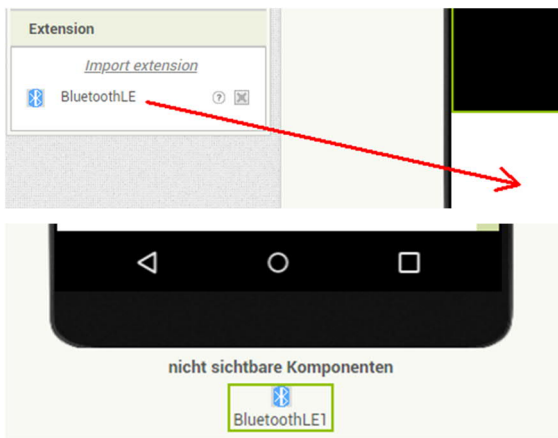


Abb. 9: Einordnung als unsichtbare Komponente

Unser Designer-Fenster sollte nun so aussehen wie in Abb. 10.

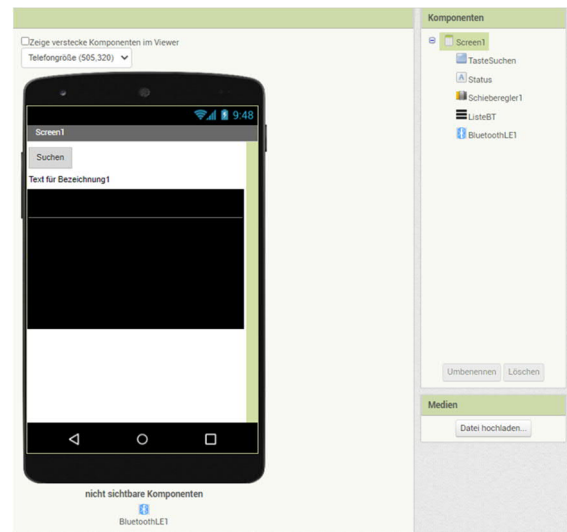


Abb. 10: Designer-Ansicht der App

Zum jetzigen Zeitpunkt haben wir nur das Aussehen und einige Eigenschaften festgelegt – nun erwecken wir die App durch die Programmierung zum Leben. Hierzu wechseln wir die Ansicht auf „Blöcke“ (rechts oben im Menu).

Zweiter Schritt: Die UUIDs festlegen

Für unser Programm benötigen wir zwei UUIDs: eine für den Dienst und die zweite für die Eigenschaften des Motors (Abb. 11).

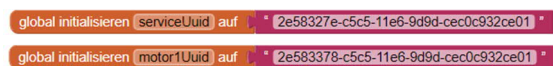


Abb. 11: UUIDs für Dienst und Motoreigenschaften

Den Block zum Initialisieren der Variablen finden wir in „Blöcke“ im Abschnitt „Eingebaut“ unter „Variablen“. Die UUID wird als Text definiert, Texte finden wir ebenfalls im Abschnitt „Eingebaut“.

Als Hilfestellung hier die beiden UUIDs als Text zum Herauskopieren:

Service-UUID:

2e58327e-c5c5-11e6-9d9d-cec0c932ce01

Motor1-UUID:

2e583378-c5c5-11e6-9d9d-cec0c932ce01

Achtung: Bei den UUIDs sollte man gewissenhaft vorgehen. Eine falsche UUID kann dazu führen, dass die App abstürzt.

Dritter Schritt: Programmablauf

Beim Klick auf die „Suchen“-Taste sollen die Bluetooth-Umgebung durchsucht und die Ergebnisse angezeigt werden. Hierzu klicken wir links in der Blöcke-Spalte auf den Eintrag „TasteSuchen“ und klicken in dem erscheinenden Menü den Eintrag „Wenn TasteSuchen.Klick mache“ an. Der Klick platziert den Block im Betrachter-Fenster (Abb. 12).



Abb. 12: Block „wenn TasteSuche.Klick“

Nun klicken wir in der Blöcke-Spalte auf den Eintrag „BluetoothLE1“ und suchen in der aufklappenden Liste nach der Funktion „StartScanning“.



Abb. 13: Funktion „StartScanning“

Diese Funktion ziehen wir in die Lücke neben „mache“ (Abb. 14).

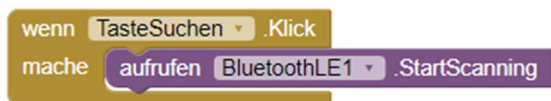


Abb. 14: Belegung von „TasteSuche“

Auf die gleiche Art und Weise wählen wir die weiteren Elemente aus und platzieren Sie nach und nach auf dem Betrachter-Fenster.

Den Block „wenn TasteSuchen.Klick“ ergänzen wir noch um eine Ausgabe für den Status (Abb. 15). *Tipp:* Das Status Element findet man unter „Blöcke“ im Abschnitt „Screen1“.



Abb. 15: Status-Feld

Sobald BT-Geräte gefunden werden, wird ebenfalls ein Ereignis ausgelöst. Dies nutzen wir, um die gefundenen Geräte in der Liste anzeigen zu lassen (Abb. 16).

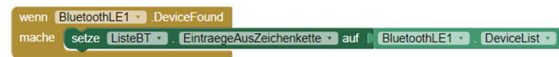


Abb. 16: Anzeige der gefundenen Geräte

In der Liste werden alle Geräte angezeigt, die unser Smartphone in der Umgebung findet. Mit einem Klick auf den Listen-Eintrag des BT-Empfängers beenden wir den Scan-Vorgang, bauen eine Verbindung auf und ändern den Text im Status. Außerdem wird die Liste unsichtbar gemacht, denn wir brauchen sie nicht mehr (Abb. 17). *Tipp:* Den Wert „falsch“ findet man unter „Logik“.

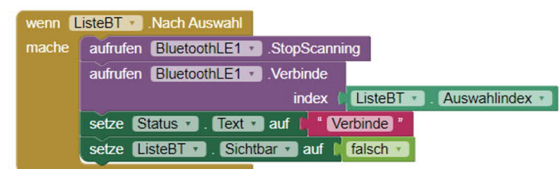


Abb. 17: Ende des Scan-Vorgangs und Verbindungsaufbau

Sobald die Verbindung aufgebaut ist, wird das Ereignis BluetoothLE1.Connected aufgerufen. Hier ändern wir den Statustext und machen den Schieberegler sichtbar (Abb. 18).



Abb. 18: Verbindung ist aufgebaut

Sobald der Schieberegler verändert wurde, wird das dazugehörige Ereignis ausgelöst. Der Schieberegler wird erst sichtbar, wenn eine Verbindung aufgebaut wurde (s. o.). Dadurch verhindert man, dass der Schieberegler versucht Daten zu senden, obwohl noch keine Verbindung mit einem Empfänger aufgebaut ist (Abb. 19).

Die Grenzwerte -128 bis +127 sind bereits im Designer hinterlegt, wobei das Vor-

zeichnen die Drehrichtung des Motors bestimmt. *Tipp:* Die Elemente für „hole“ findet man unter „Blöcke“ und „Variablen“.

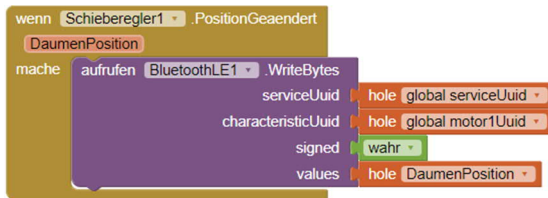


Abb. 19: Funktion des Schiebereglers

Verbindung und Test des Programms

Sehr bequem funktioniert die Verbindung über WLAN; üblicherweise befindet sich das Smartphone im gleichen lokalen Netzwerk wie der PC (und dies ist auch die Voraussetzung dafür, dass die Verbindung aufgebaut wird).

Auf dem Smartphone starten wir den „MIT AI2 Companion“. Möglicherweise fragt die App nach Zugriffsrechten; diese müssen wir zulassen. Auf dem PC wählen wir im Menü „Verbinden“ die Option „AI Companion“ (Abb. 20).

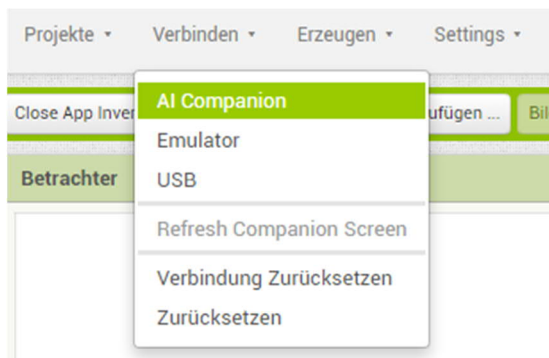


Abb. 20: Start des MIT AI2 Companion

Nun wird ein Fenster mit einem QR-Code und einem 6-stelligen Zeichencode angezeigt – durch Klicken auf „Scan QR Code“ auf dem Smartphone wird die Verbindung zum PC aufgebaut und nach einiger Zeit erscheint das Design unseres Programms auf dem Bildschirm.

Für den ersten Test schließen wir einen Motor an M1 des BT-Empfängers an und

versorgen ihn mit Spannung. Nicht vergessen: Den roten Knopf des Empfängers so lange drücken, bis die blaue LED schnell blinkt.

Sobald wir auf dem Smartphone die „Suchen“ Schaltfläche gedrückt haben, sollte sich die Liste mit den gefunden Geräten aufbauen. Sie ist sortiert nach Empfangsstärke, d. h. das Gerät mit dem besten Empfang steht oben. Der BT-Empfänger wird als „BT Control Receiver“ angezeigt – diesen Eintrag wählen wir durch Antippen aus.

Tipp: Bei manchen Smartphones/Tablets kann die Funktionalität von BT mit anderen Sensoren gekoppelt sein. Beim Beta-Test stellte sich heraus, dass neben BT auch GPS eingeschaltet werden musste.

Daraufhin sollte sich die Liste schließen und der Schieberegler erscheinen – und man kann nun mit dem Schieberegler den Motor in unterschiedlichen Geschwindigkeiten laufen lassen.

Das Programm ist sehr stark vereinfacht; es werden keine Fehler abgefangen, und wenn die Verbindung abbricht, muss man es z. B. durch Auswahl von „Verbinden → Refresh Companion Screen“ einmal neu starten. Das werden wir im nächsten Schritt verbessern; außerdem werden wir die restlichen Motoren und den Servo ansteuern.

Kosmetik

Im Folgenden werden wir uns mit den Möglichkeiten beschäftigen, das Aussehen unserer App etwas ansprechender zu gestalten.

Tipp: Zwar kann man im Browser im Betrachter-Fenster sehen, welche Elemente man platziert und wohin – aber später auf dem Smartphone/Tablet sieht es doch sehr viel anders aus. Wenn man sich mit dem MIT AI2 Companion verbindet, kann man auf dem Bildschirm des Smartphone live mitverfolgen, wie das Layout aussehen wird.

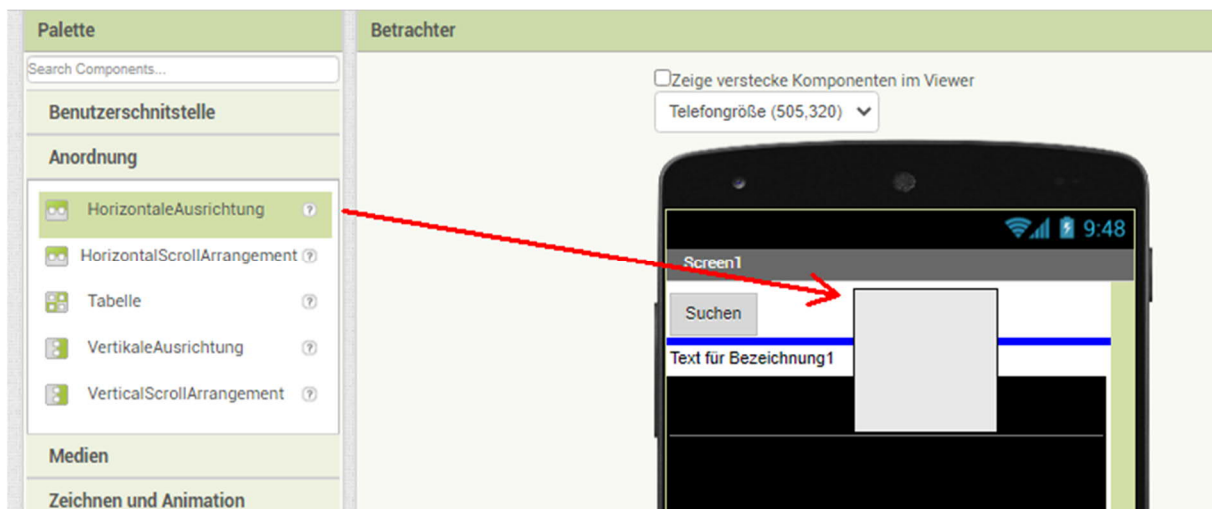


Abb. 21: Einstellung der horizontalen Ausrichtung

Zunächst wechseln wir wieder zum Designer (Schaltfläche rechts oben). In der Spalte Palette klicken wir „Anordnung“ an. Wir ziehen eine „horizontale Ausrichtung“ direkt unter die Taste „Suchen“ (Abb. 21).

Tipp: An der blauen Linie kann man erkennen, wo das neue Element platziert wird. Bei Bedarf kann man die Elemente aber auch nachträglich noch verschieben.

Bei den Eigenschaften der horizontalen Ausrichtung ändern wir die Ausrichtung-Horizontal auf „Mitte: 3“ und die Breite auf „Fuelle alles“. Nun ziehen wir die Taste „Suchen“ mit der Maus in die horizontale Ausrichtung hinein (Abb. 22).

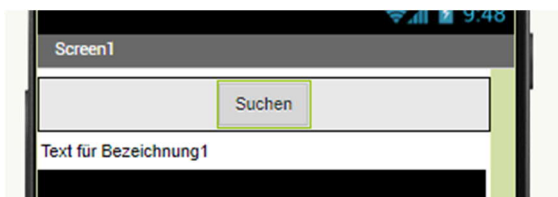


Abb. 22: Taste „Suchen“ horizontal ausgerichtet

Anschließend wählen wir wieder in der Palette den Bereich Benutzerschnittstelle und ziehen eine Taste rechts neben die Taste „Suchen“. Diese Taste benennen wir in der Spalte „Komponenten“ auf „TasteReset“ um; in den Eigenschaften ändern wir den Text auf „Reset“ (Abb. 23).



Abb. 23: Taste „Reset“

Nun kommen wir zum kompliziertesten Teil: die insgesamt vier Schieberegler. Zunächst machen wir den bestehenden Schieberegler sichtbar, das heißt wir setzen in den Eigenschaften den entsprechenden Haken.

Aus der Palette → Anordnung ziehen wir uns eine „Tabelle“ zwischen Schieberegler und Listenansicht. Bei den Eigenschaften der Tabelle ändern wir die Breite auf „Fuelle alles“ und die Anzahl der Zeilen auf 10.

Den Schieberegler ziehen wir nun in die Tabelle hinein, möglichst auf die zweite Zeile und die rechte Spalte (blaues Rechteck wird angezeigt). Sollte man die richtige Zeile oder Spalte nicht treffen, kann das jederzeit geändert werden.

Anschließen ziehen wir uns drei weitere Schieberegler in die Tabelle, immer mit einer Zeile Abstand und in die rechte Spalte. Bei diesen neuen Schieberegler ändern wir in den Eigenschaften den „Maximalwert“ auf 127 und den „KleinsterWert“ auf -127.

Danach ziehen wir viermal eine „Bezeichnung“ in die Zeilen 1, 3, 5 und 7 (jeweils oberhalb bzw. unterhalb der Schieberegler). In den Eigenschaften ändern wir den Text auf „Motor 1“, „Motor 2“, „Motor 3“ und „Servo“.

In die unterste Zeile (Zeile 10) ziehen wir eine Taste, die wir in der Komponentenspalte auf „TasteStopp“ umbenennen, und in den Eigenschaften ändern wir den Text auf „Stopp“. Diese Taste nutzen wir später, um sämtliche Motoren abzuschalten, denn es ist nicht immer ganz einfach, den Mittelpunkt des Schiebereglers zu finden.

Zuletzt ziehen wir uns eine Bezeichnung in die Tabelle in die erste (!) Spalte und in der

9. Zeile (zwischen dem Schieberegler für den Servo und der Stopp-Taste).

Den Text in den Eigenschaften löschen wir, die Breite setzen wir auf „25 percent“ und die Höhe auf „5 percent“. Hier wird die Bezeichnung genutzt, um die Tabelle horizontal mittig auszurichten und eine Leerzeile zwischen Schieberegler und Stopp-Taste zu erzeugen. Nun sollte das Design im Betrachter so aussehen wie in Abb. 24.

Bevor wir zur Programmierung übergehen, wählen wir bei den Komponenten die „Tabelle1“ aus und löschen in den Eigenschaften den Haken „sichtbar“. Dadurch werden alle Schieberegler gleichzeitig unsichtbar.

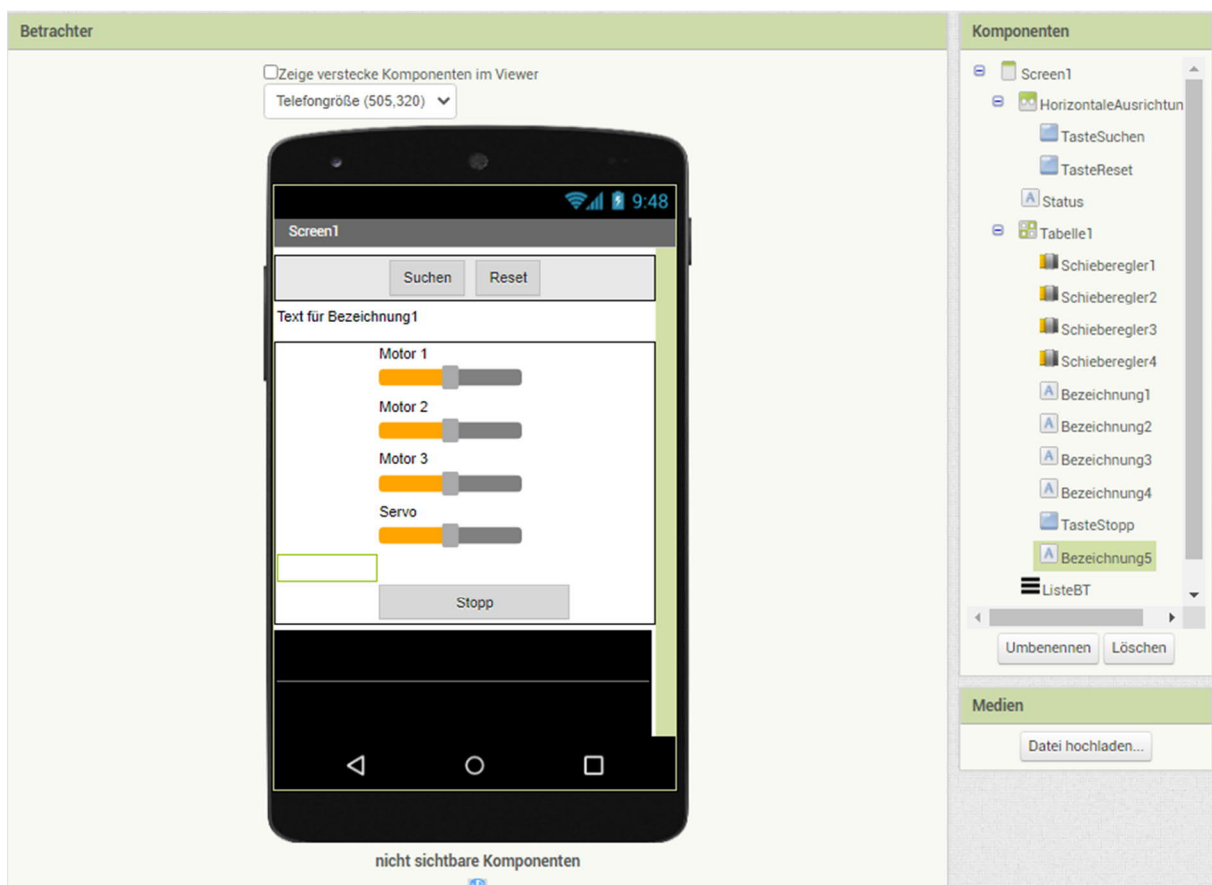


Abb. 24: Anordnung und Konfiguration der Schieberegler

Programmänderung

Zunächst brauchen wir noch die UUIDs für die restlichen Motoren und den Servo (Abb. 25).

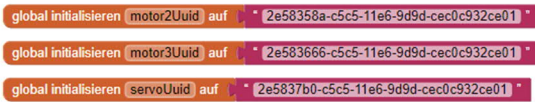


Abb. 25: UUIDs für Motoren und Servo

Bei der Eingabe können wir zwei Tricks anwenden: Durch Rechtsklick auf die vorhandene „motor1Uuid“ klappt ein Menü auf, in dem wir „Duplizieren“ auswählen. Das machen wir dreimal, anschließend müssen wir nur die automatisch generierten Variablennamen abändern (Abb. 26).



Abb. 26: Duplizieren und Anpassen der UUIDs

Die UUIDs unterscheiden sich nur in den letzten vier Ziffern der ersten Gruppe (Abb. 27).



Abb. 27: Anpassung der duplizierten UUIDs

Auch hier als Hilfestellung die UUIDs als Text zum Kopieren:

Motor2-UUID:
2e58358a-c5c5-11e6-9d9d-cec0c932ce01

Motor3-UUID:
2e583666-c5c5-11e6-9d9d-cec0c932ce01

Servo-UUID:
2e5837b0-c5c5-11e6-9d9d-cec0c932ce01

Das Ereignis „BluetoothLE1.Connected“ ändern wir wie in Abb. 28 gezeigt.



Abb. 28: Änderung des Ereignisses „BluetoothLE1.Connected“

Tipp: Rechter Mausklick auf „setze Schieberegler1.Sichtbar“ und dann im Menü „2 Blöcke löschen“. Anschließend im Bereich „Tabelle1“ den oben gezeigten Block einfügen.

Nun müssen wir die Ereignisse für die hinzugekommenen Schieberegler definieren. Hier können wir ebenfalls „duplizieren“ anwenden (rechter Mausklick auf den „Position geändert“-Block).

Da nach dem Duplizieren zwei Blöcke mit gleichem Namen existieren, wird ein rotes Kreuz im Block links oben angezeigt. Mit Klick auf das kleine Dreieck neben dem Namen bekommen wir eine Liste mit den möglichen Namen angezeigt – hier suchen wir nacheinander die passenden Namen aus.

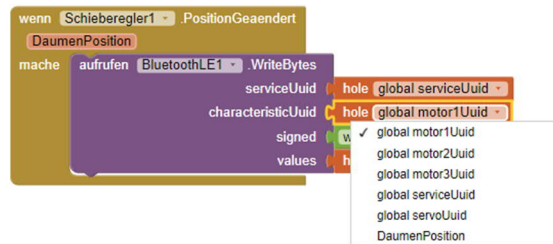


Abb. 29: Auswahl der richtigen „CharacteristicUUID“

Anschließend müssen wir bei den drei neuen Schieberegler-Ereignissen die „CharacteristicUuid“ richtig zuordnen (Abb. 29). Auch hier können wir mit dem kleinen Dreieck eine Liste aufklappen und die UUIDs zuordnen.

Tipp: Nicht die serviceUuid mit der servoUuid verwechseln.

Danach fügen wir die Ereignisse für die neuen Tasten hinzu (Abb. 30).

Tipp: Die „wenn dann“ Verzweigung findet man im Bereich „Eingebaut“ unter „Steuerung“, die Zahl 0 unter „Mathematik“.

Nun fehlt nur noch die Verbindung über den MIT AI2 Companion (s. o.) und wir können das Programm testen.

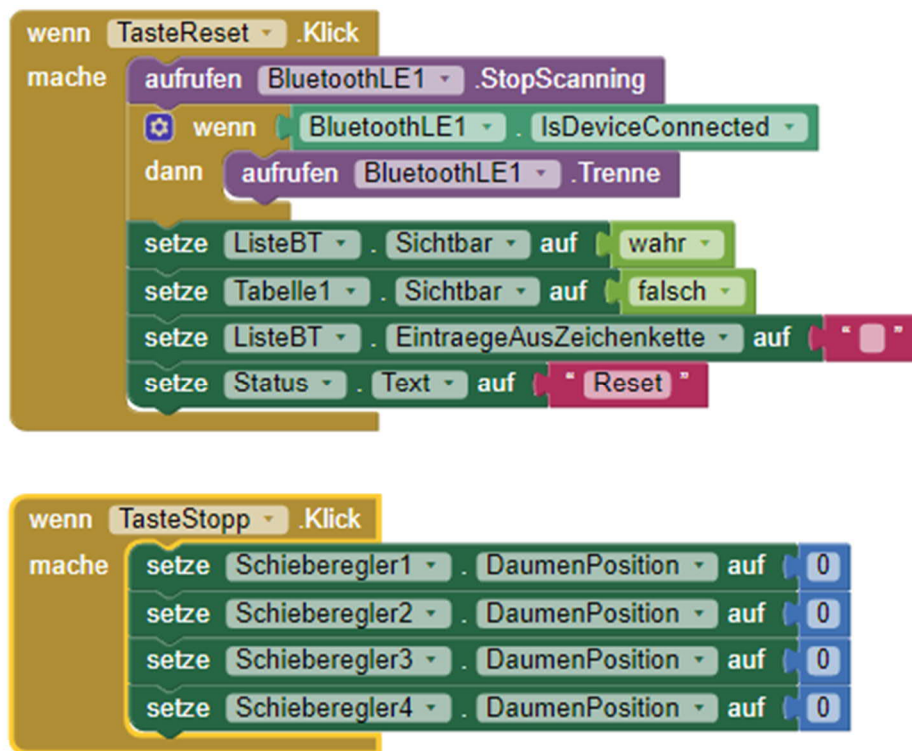


Abb. 30: Zuordnung der Ereignisse zu den beiden Tasten

Aussichten

Diese Einführung zeigt nur einen Bruchteil der Möglichkeiten des App Inventors. Da gibt es noch sehr viel Potential:

- Nutzung der eingebauten Beschleunigungssensoren, so dass man nur durch Bewegung des Smartphones steuert
- Steuerung mittels virtuellem Joystick – wie bei der fischertechnik-App
- zeitgesteuerte Sequenzen, um automatische Vorgänge zu steuern
- Ansteuerung anderer Geräte, z. B. des BT Smart Controllers

... und vieles mehr. Die Bluetooth-LE Extension bietet auch die Möglichkeit, die UUIDs selbst auszulesen, sodass man sich das Eingeben ersparen könnte. Ich habe in diesem Rahmen darauf verzichtet, weil das Programm für einen Anfänger schlecht nachvollziehbar und ziemlich unübersichtlich ist. Wer möchte, kann es sich aber [hier](#) anschauen (Datei [GATT_Test.aia](#)).

Danksagung

Bedanken möchte ich mich bei Thomas und Tim Brestrich, die den Artikel geprüft und mir wertvolle Hinweise gegeben haben.

Anhang: Wie man die UUIDs herausfindet

Hierzu verwendete ich die App „nRF Connect for Mobile“ von Nordic Semiconductor (Abb. 31, zu finden im Play Store), die relativ bequem BT-Geräte auflistet und auch die Möglichkeit anbietet, das GATT-Profil auszulesen.

Verbindet man diese App mit dem BT-Empfänger, erhält man zunächst eine Übersicht der Dienste – auf den ersten Blick viele Zahlen (Abb. 32, links). Aber durch einfaches Durchprobieren kommt man schließlich zum Ergebnis, dass sich hinter dem untersten „Unknown Service“ die Eigenschaften verbergen, die für die Nutzung des BT-Empfängers interessant sind (Abb. 32).

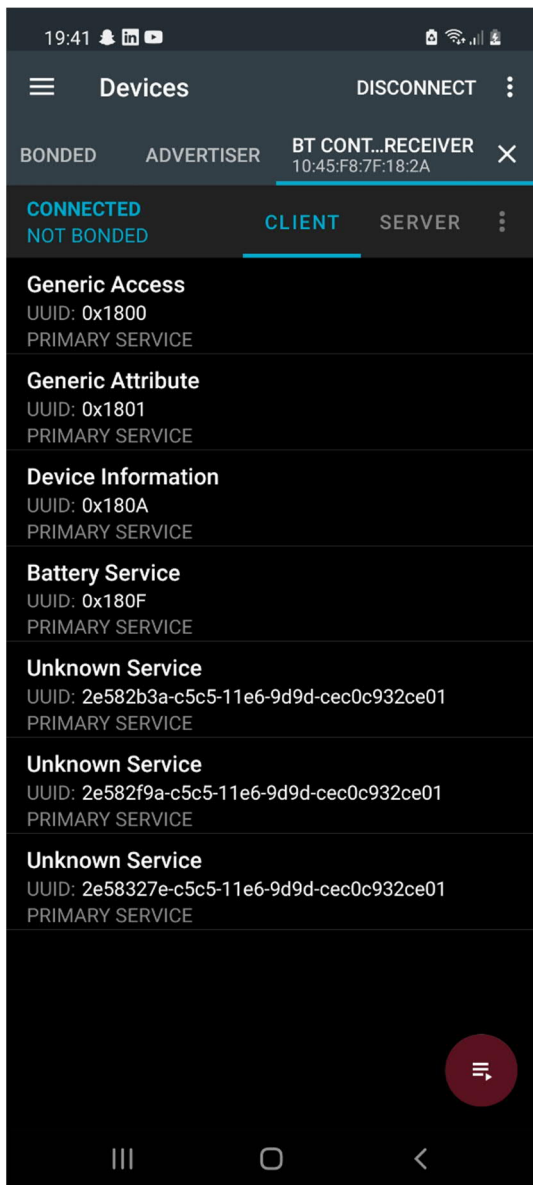


Abb. 31: App nRF Connect for Mobile

Die obigen vier Eigenschaften (Characteristics) können jeweils mit Bytes beschrieben und gelesen werden. Durch Versuche findet man heraus, dass sie zu den drei Motoren und dem Servo gehören (von oben nach unten: M1, M2, M3 und Servo).



Abb. 32: „Characteristics“ der vier Motoren
Gesendet wird ein Byte mit Vorzeichen, d. h. wir haben einen Wertebereich von -128 bis +127.

Computing

TX-Light: Arduino (Uno/Mega) und ftDuino aus ROBO Pro ansteuern

Holger Howey

Die von fischertechnik mit der Einführung des TXT 4.0 abgekündigte Programmiersprache ROBO Pro ist nach wie vor eine leistungsfähige Einsteigerprogrammiersprache. Leider unterstützt sie keine „Fremdinterfaces“ wie z. B. den in Schulen verbreiteten Arduino. Doch auch für diese Herausforderung hat die Community eine Lösung...

Hintergrund

Ausgangspunkt für die Suche nach einer Möglichkeit, Arduinos mit ROBO Pro zu programmieren, war, dass in unserer Schule zwar älteres fischertechnik vorhanden war, aber die Interfaces nicht an aktuelle Rechner angeschlossen werden konnten. Arduino Unos waren aber vorhanden.

Dabei bin ich im Forum der fischertechnik-Community auf das Projekt „fx1-arduinoparser“ (= Syntax-Analysierer) von mrkubikus gestoßen [1]. Er hat über die Auswertung der USB-Signale und die im Netz auffindbaren Schnittstellen-Dokumentationen ein Programm entwickelt, mit dem man aus ROBO Pro den Arduino Uno im Online-Modus ansteuern kann.

Leider war keine weitere Beschreibung über die Nutzung verfügbar, und diejenigen, die den Parser eingesetzt haben, sind an ein paar Kleinigkeiten hängen geblieben.

Außerdem ist es nicht möglich, an den Arduino direkt Motoren oder Lampen anzuschließen; dazu benötigt man so genannte H-Brücken [2]. Für Menschen, die sich nicht damit auskennen, ist spätestens hier Schluss, da man die Treiberbausteine hinzukaufen muss.

Auch ist kein „Timeout“ vorhanden. Ein solcher „Watchdog“ (= Wachhund) sorgt

dafür, dass die Motoren bei Programmende, bei Störung der USB-Übertragung oder wenn das USB-Kabel abgezogen wird, abgeschaltet werden.

Ich habe beschlossen, dieses Programm als Grundlage zu nehmen und den Kommunikationsteil daraus zu verwenden. Das Schalten und das Einlesen der I/O-Pins des Arduino waren bereits enthalten.

Mit weiteren Funktionen gab es jedoch viele Schwierigkeiten. Das lag u. a. daran, dass zum Zeitpunkt der Programmierung des Parsers ROBO Pro weiterentwickelt wurde. Übertragene Variablen standen an anderen Stellen, es kamen ganz neue hinzu oder die Datenlänge veränderte sich etc. pp.

Die Beschreibungen zur Programmierung der fischertechnik-Interfaces (wie z. B. des TX Controllers) in C oder C++ gehen davon aus, dass ein Programm das Interface vom PC aus steuert, und nicht auf der Empfängerseite die Daten des PC auswertet.

Ich musste sehr vieles ausprobieren, den Datenstrom analysieren und dann das Programm anpassen. Das war nicht einfach. Im Prinzip schreibt der Parser in einen Speicherbereich, über den eine Tabelle mit Variablen gelegt wird. Das muss man erst einmal verstehen – und wenn Fehler auftreten, diese auch finden.

TX-Light

Aus dem „fx1-arduino-parser“ entstand so „TX-Light“: ein Arduino-Sketch, der via USB-Kabel Kommandos von ROBO Pro empfängt, auswertet und die entsprechenden Anschlüsse (Pins) des Arduino ausliest (Input) oder die gewünschte Spannung anlegt (Output).

Dazu musste ich zunächst ein Kommunikationsprotokoll festlegen, denn ROBO Pro unterstützt die Kommunikation mit unterschiedlichen fischertechnik-Interfaces, die jeweils abweichende Protokolle verwenden. Ich habe die Schnittstelle zum TX gewählt, weil die seit vielen Jahren stabil ist und nach Einführung des TXT nicht mehr weiterentwickelt wurde. Das Protokoll (und damit mein Programm „TX-Light“) bleibt daher auch in neueren Versionen von ROBO Pro dasselbe.

Motorausgänge M1 bis M4

Man kann fischertechnik-Motoren mit dem Arduino nicht direkt ansteuern, da die Ports nur 5 V liefern und eine maximale Last von 40 mA verkraften. Daher muss man entweder Motortreiber (H-Brücken) verwenden, die dann jeweils einen Pin belegen, oder aber ein aufsteckbares „Motor Shield“ (eine Platine) mit Motortreibern erwerben. Bewährt hat sich das Motor Shield v2.3 von Adafruit, da es keinen der Pins des Arduino belegt, sondern die Steuerung der (insgesamt vier) Motorausgänge über das I2C-Protokoll vornimmt. Es kann mit bis zu 12 V versorgt werden und verträgt eine Last von bis zu 1,2 A je Motorausgang – ausreichend für unsere fischertechnik-Motoren. Es können sogar mehrere Adafruit Motor Shields auf einen Arduino aufgesteckt („kaskadiert“) werden, und über den Arduino-Treiber lassen sich auch Schrittmotoren und Servos ansteuern.

Im TX-Light habe ich die in der folgenden Tabelle gezeigte intuitive Zuordnung der Motorausgänge vorgenommen.

ROBO Pro	Arduino	Motor Shield	Funktion
M1	./.	M1	Motorausgang
M2	./.	M2	Motorausgang
M3	./.	M3	Motorausgang
M4	./.	M4	Motorausgang

Tab. 1: Motorausgänge M1 bis M4

Damit lassen sich auch die Geschwindigkeiten, die ROBO Pro überträgt, im Programm auswerten und an den Motortreiber weitergeben.

Ausgänge O1 bis O8

Die acht O-Ausgänge hatte mr-kubikus bereits auf die Pins 6-13 gelegt. Nicht an jedem Ausgang kann die Spannung über ein pulsweitenmoduliertes Signal (PWM) reguliert werden; an den digitalen Ausgängen ist nur die Einstellung von „Ein“ (5V) und „Aus“ (0V) möglich.

Achtung: Diese Ausgänge dürfen maximal mit je 40 mA und insgesamt nicht mehr als 200 mA belastet werden!

Tabelle 2 zeigt die Zuordnung in einer Übersicht.

ROBO Pro	Arduino	Motor Shield	Funktion
O1	Pin 6	./.	PWM-Ausgang
O2	Pin 7	./.	digitaler Ausgang
O3	Pin 8	./.	digitaler Ausgang
O4	Pin 9	./.	PWM-Ausgang
O5	Pin10	./.	PWM-Ausgang
O6	Pin 11	./.	PWM-Ausgang

ROBO Pro	Arduino	Motor Shield	Funktion
O7	Pin 12	./.	digitaler Ausgang
O8	Pin 13	./:	digitaler Ausgang

Tab. 2: O-Ausgänge O1-O8

Mit der SoftPWM-Bibliothek könnte man alle Output-Pins zu PWM-Ausgängen machen, allerdings kommt da der Uno an seine Grenzen.

Eingänge I1 bis I8

Einige Pins des Arduino sind – anders als beim fischertechnik ROBO TX Controller und ähnlich den früheren fischertechnik Interfaces – nur als digitaler Eingang (Pin 0 bis 13) nutzbar. Direkte EX- und EY-Eingänge wie beim parallelen fischertechnik-Interface sind nicht vorhanden. Stattdessen kann man die analogen Eingänge A0 bis A5 verwenden.

Doch einige der Pins sind bereits belegt: Über Pin 0 und Pin 1 erfolgt die serielle Übertragung zum PC (ROBO Pro), und die Pins A4 und A5 benötigt der I2C-Bus (Ansteuerung des Adafruit Motor Shield). Die Pins 6 bis 13 sind bereits für die O-Ausgänge belegt.

Somit kann man nur vier der digitalen (Pin 2 bis Pin 5) und fünf der analogen Pins (A0 bis A4) als Eingang nutzen. Der Sketch „TX-Light“ nimmt die in Tabelle 3 gelistete Zuordnung vor.

ROBO Pro	Arduino Uno	Funktion
I1	Pin 2	digitaler Eingang
I2	Pin 3	digitaler Eingang
I3	Pin 4	digitaler Eingang
I4	Pin 5	digitaler Eingang
I5	Pin A0	digitaler / analoger Eingang

ROBO Pro	Arduino Uno	Funktion
I6	Pin A1	digitaler / analoger Eingang
I7	Pin A2	digitaler / analoger Eingang
I8	Pin A3	digitaler / analoger Eingang

Tab. 3: Eingänge I1 bis I8

Abb. 1 zeigt die Ein- und Ausgänge an den Pins des Adafruit Motor Shield.

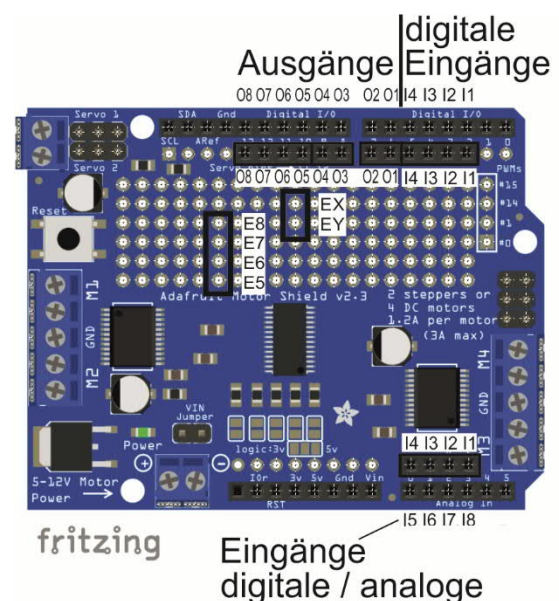


Abb. 1: Ein- und Ausgänge am Arduino (Adafruit Motor Shield)

Auch die Eingänge hab ich unempfindlicher gegen Störstrahlung (z. B. vom Monitor) gemacht und eine Buchse auf das Motor Shield aufgelötet. Die Belegung entspricht dem alten parallelen fischertechnik-Interface: Man kann also die alten Modelle direkt über das Flachbandkabel verbinden und mit ROBO Pro ansteuern. Wie das geht, hat Peter Gabriel vor zwei Jahren in der ft:pedia vorgestellt [3]. Außerdem gibt es eine ausführliche bebilderte Anleitung zum Download, nach der man den Anschluss selber ergänzen kann.

Abb. 2 zeigt einen Arduino Uno mit Motor Shield nach dem „Umbau“, der einen originalen fischertechnik-Roboter-Bausatz aus dem Baukasten *Computing Experimental* von 1987 ([30573](#)) steuert.

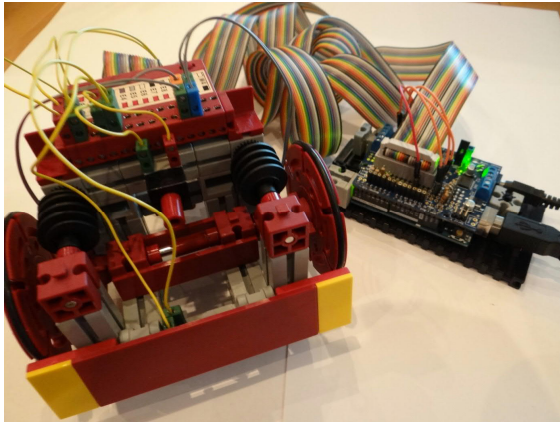


Abb. 2: Arduino Uno mit Adafruit Motor Shield und Turtle aus *Computing Experimental*

Sehr schön ist auch, dass der Interface-Test von ROBO Pro die an den Eingangs-Pins anliegenden Werte anzeigt – das kann die Arduino-IDE nicht.

TX-Light für den ftDuino

Der nächste Schritt war dann die Unterstützung des ftDuino [4]. Am ftDuino ist kein Motor Shield erforderlich: Der ftDuino basiert auf einem Arduino Leonardo und verfügt bereits über Motortreiber. Er ist dafür gedacht, fischertechnik-Modelle mit einem kurzschlussfesten, mit fischertechnik-Anschlüssen und -Sensoren kompatiblen Arduino anzusteuern. In der Anleitung findet man viele Beispiele dafür, wie man über die Arduino IDE Eingänge einlesen und Ausgänge steuern kann [5]. Dazu bietet der ftDuino eine Erweiterung der Arduino IDE an, die man nutzen kann, um z. B. die Geschwindigkeit eines Motors einzustellen.

Es kommt vor, dass Käufer den ftDuino als billigen Ersatz für einen TX, TXT oder TXT 4.0 missverstehen und erwarten, dass er mit ROBO Pro oder ROBO Pro Light angesteuert werden kann. Wird er dort nicht erkannt, wird reklamiert...

Daher habe ich den Arduino-Sketch „TX-Light“ für den ftDuino umgeschrieben. Damit kann der ftDuino nun auch als „TX“ aus ROBO Pro gesteuert werden. Dabei gibt es ein paar Besonderheiten:

- beim ftDuino sind alle Ausgänge mit PWM ansteuerbar
- der ftDuino hat vier schnelle Zählereingänge, an die (wie beim TX) die Geber der fischertechnik-Encodermotoren angeschlossen werden können.

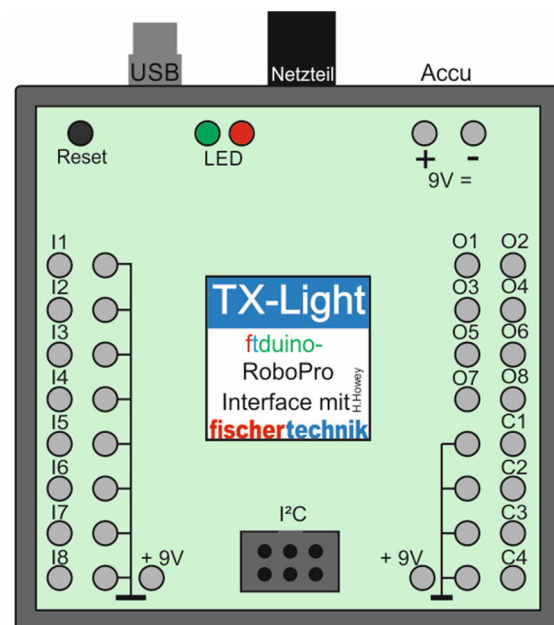


Abb. 3: Anschlüsse des ftDuino

Installation

Die Installation für Arduino und ftDuino ist einfach: Es muss nur die entsprechende INO-Datei auf den Arduino bzw. ftDuino übertragen werden, dann kann man mit ROBO Pro loslegen.

Das Grove/Seeed-System

Es ist auch möglich mit dem Grove/Seeed-System zu arbeiten und dessen Anschlüsse aus ROBO Pro zu nutzen. Das Baseboard wird dazu „huckepack“ auf den Arduino aufgesteckt.

Abb. 4 zeigt die Belegung der Ein- und Ausgänge des Grove/Seeed Baseboard.

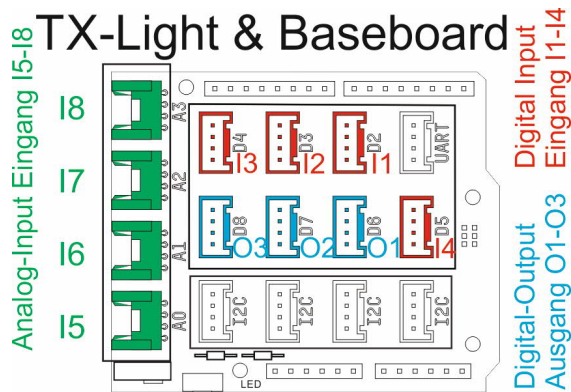


Abb. 4: Anschlussbelegung bei Grove/Seed

Wie einfach die Nutzung ist, zeigt das Beispiel in Abb. 5. Dort ist eine LED an den Ausgang O1 eines Seed Baseboards angeschlossen.

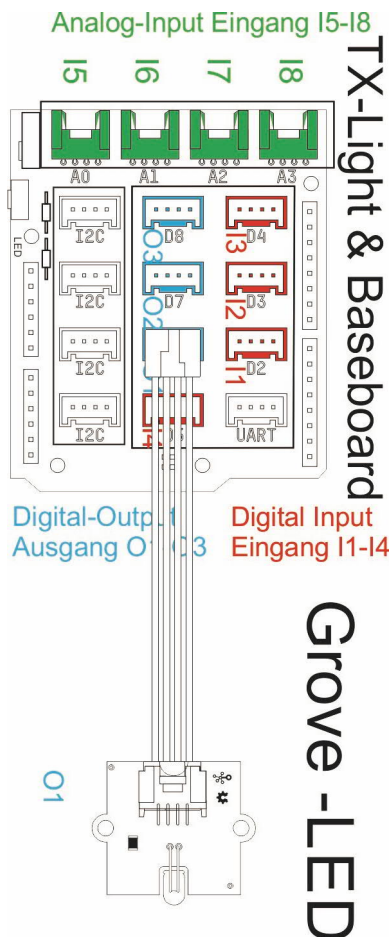


Abb. 5: LED, angeschlossen an O1 des Grove/Seed Baseboards

Das ROBO Pro-Programm, das die so angeschlossene LED blinken lässt, zeigt Abb. 6.

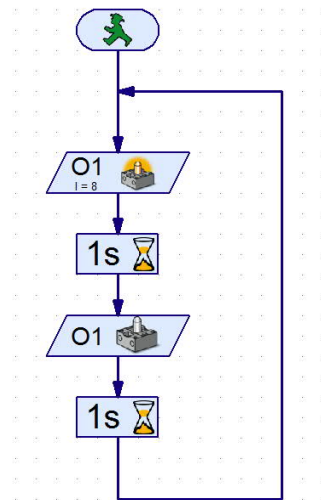


Abb. 6: LED blinkt an O1

Weitere Aussichten

Wer programmieren kann, kann statt der normalen fischertechnik-Motoren auch vier Schrittmotoren an M1-M4 zweier „gestapelter“ Motor Shields ansteuern. Dafür muss das untere Motor Shield mit *Stacking Headers* ausgestattet werden.

Es ist auch möglich alte fischertechnik-Interfaces als Treiber zu nutzen: Entweder direkt, indem man das IC auf dem Interface entfernt und den Pin vom Arduino mit der H-Brücke verbindet, oder indem man selbst das ganze Interface ansteuert. Das ist bisher im „TX-Light“-Sketch nicht vorgesehen.

Fazit

ROBO Pro ist eine hervorragende Programmierumgebung. Man kann sie, mit kleinen Einschränkungen, mit einem Arduino Uno oder einem ftDuino nutzen. Dazu muss man lediglich das Programm „TX-Light“ (INO-Datei) aufspielen. Wer möchte, kann sich selber ein preiswertes Interface bauen, um damit alte fischertechnik-Modelle wieder zum Leben zu erwecken oder Neues auszuprobieren.

Die Dateien können im [Downloadbereich dieser Ausgabe](#) der ft:pedia heruntergeladen werden.

Referenzen

- [1] mr-kubikus: [fx1-arduino-parser](#). V0.3, 11.08.2016. Github.com.
- [2] Tomas Magin: *Die Kunst der H-Brücke: Let's Rock*. ft:pedia 2/2022, in dieser Ausgabe.
- [3] Peter Gabriel: *Adapter für fischertechnik-Computing-Modelle*. [ft:pedia 1/2020](#), S. 75–78.
- [4] Till Harbaum: *ftDuino – Open-Source trifft Konstruktionsbaukasten*. [ft:pedia 1/2018](#), S. 85–91.
- [5] Till Harbaum: [ftDuino – ein fischertechnik kompatibler Arduino](#). Bedienungsanleitung. 12.01.2021.

Computing

TX-Simulator für den ftDuino

Dirk Fox

Kaum hatte fischertechnik die Weiterentwicklung von ROBO Pro abgekündigt, machte die Community den TXT 4.0 ROBO Pro-kompatibel [1]. Jetzt kann auch der ftDuino mit ROBO Pro gesteuert werden, als wäre er ein TX: Angeregt von Holger Howes Erweiterung des fx1parsers [6, 10, 11] entstand ein „TX-Simulator“, der auch das I2C-Protokoll beherrscht.

Das Ende von ROBO Pro?

Mit der Ankündigung von fischertechnik, dass ROBO Pro nicht mehr an den TXT 4.0 angepasst wird, wurde die grafische Programmiersprache von fischertechnik Mitte 2021 schlagartig zum „Auslaufmodell“. Auch wenn ROBO Pro immer ein fischertechnik-Sonderweg war, so hat die Sprache doch zahlreiche unbestrittene Stärken, wie das Multi-Threading quasi-paralleler Programme, die vom Programmfluss unabhängigen Datenflüsse oder die geniale Unterstützung der Programmierung endlicher Automaten in Gestalt von Zustandsübergangsdiagrammen [2].

ROBO Pro ist über Jahrzehnte gereift [3], daher ist die Zahl der bekannten Bugs inzwischen sehr klein. Und schließlich gibt es für den TX und den TXT (gar nicht zu sprechen vom ROBO Interface) keine andere (Einsteiger-) Programmiersprache – abgesehen von der Community-Firmware, mit der zumindest der TXT auch via Brickly [4] oder StartIDE [5] programmiert werden kann. Kein Wunder, dass der Aufschrei in der Community angesichts der Abkündigung groß war. Und prompt „bäumte“ sich die Fangemeinde auf – und gebar den ftrobopy-Server [1], über den nun auch der neue TXT 4.0 mit ROBO Pro (im Online-Mode) gesteuert werden kann. Das funktioniert via WLAN oder Bluetooth sogar kabellos – z. B. für mobile Roboter.

ROBO Pro für den Arduino

In diesem Kontext bekam auch das Fan-Projekt von ft-ninja (alias mr-kubikus) neuen Rückenwind. Im Dezember 2012 hatte dieser einen Parser für den Arduino entwickelt, der sich gegenüber ROBO Pro als TX ausgibt. Darüber kann ein Arduino im Online-Mode mit ROBO Pro gesteuert werden [6]. Zuletzt veröffentlichte er am 11.08.2016 die Version 0.3 des Parsers auf Github [7] und testete sie am 19.02.2021 mit ROBO Pro 4.6.6 [6].

Die Lösung unterliegt allerdings verschiedenen technischen Einschränkungen, denn die Eingänge des Arduino vertragen maximal 5 V, während der TX(T) bis zu 10 V aushält. Auch lassen sich die Ports des Arduino nicht universell nutzen wie beim TX(T), sondern nur entweder als analoger (6 Ports) oder als digitaler Eingang (14 Ports).

Der Betrieb von fischertechnik-Motoren an den Ports ist nur mit ergänzenden H-Brücken [8] möglich, da die Ausgänge des Arduino mit maximal 20 mA belastet werden können und nicht mehr als 5 V liefern – das reicht bestenfalls für eine LED. Schließt man einen Motor an, der leicht 200 mA Strom zieht, steigt Rauch auf...

Zudem ist keiner der Eingänge kurzschlussfest oder gegen Überlast gesichert – eine

Hauptursache für die kurze Lebenserwartung von Arduinos in Kinderzimmern und Schulen.

Für die Bereitstellung von vier schnellen Zählereingängen benötigt man vier unabhängige Interrupt-Service-Routinen (ISR) – der Arduino (Uno) kennt aber lediglich zwei Interrupts, die über die Ports D2 und D3 eine ISR auslösen können. Mehr als zwei schnelle Zählereingänge sind mit einem Arduino (Uno) also grundsätzlich nicht realisierbar. Im Arduino-Parser sind sie gar nicht vorgesehen. Schließlich kann der fischertechnik-Ultraschall-Abstandssensor nicht genutzt werden, und auch die I2C-Befehle von ROBO Pro werden nicht unterstützt.

Eine naheliegende Verbesserung ist die Ergänzung des Arduino um ein Adafruit Motor Shield (v2.3) (Abb. 1 [9]). Dessen vier PWM-Motor-Ausgänge können mit bis zu 12 V und 1,2 A belastet werden. Eine entsprechende Anpassung des von ft-ninja „fx1parser“ getauften Sketches an einen Arduino mit Adafruit Motor Shield hat Holger Howey in seinem „TX-Light“-Projekt 2017 umgesetzt [10, 11].

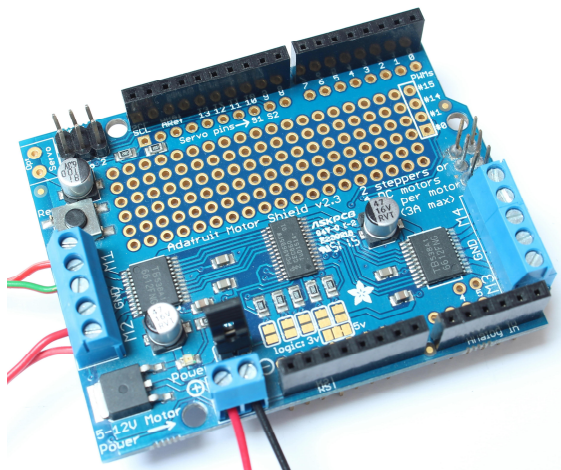


Abb. 1: Adafruit Motor Shield v2.3 [9]

Allerdings belegt das Shield zwei der sechs analogen Ports (A4, A5) für das I2C-Protokoll, über das es vom Arduino angesteuert wird – so bleiben maximal vier Ports, die als analoger Eingang genutzt werden können.

Alternative: Der ftDuino

Gegen die angeführten Beschränkungen des Arduino, der sich vor allem wegen der Fülle an Treibern und Maker-Projekten gerade für den Schulunterricht besonders eignet, ist Till Harbaum 2018 mit der Entwicklung des ftDuino angetreten, der auf dem Design des Arduino Leonardo basiert [12, 13].

Mit dem ftDuino kann man mit nur wenigen Einschränkungen einen TX simulieren, denn sein Konzept und Layout wurden an zentralen Leistungsmerkmalen des TX ausgerichtet:

- Alle Input-Ports lassen sich wie beim TX sowohl digital als auch analog nutzen.
- Die Eingänge sind kurzschluss- und überspannungsgesichert; sie messen Spannungen von bis zu 10 V.
- Widerstandsmessungen bis 10 kOhm sind mit einer Auflösung von 10 bit möglich (TX: 5 kOhm, Auflösung: 12 bit).
- Die Interrupts des ftDuino reichen für vier schnelle Zählereingänge, sodass der ftDuino wie der TX vier Encodermotoren steuern kann.
- Die Motorausgänge des ftDuino sind mit bis zu 600 mA (insgesamt mit 1,2 A) belastbar; das ist mehr, als fischertechnik für den TX garantiert (250 mA je Motorausgang und 1 A gesamt).
- Der Wannenstecker des ftDuino zum Anschluss von I2C-Sensoren und -Aktoren ist Pin-kompatibel mit dem EXT2-Anschluss des TX, über den dessen I2C-Bus erreichbar ist. Auch der I2C-Bus des ftDuino arbeitet mit 5V-Logik. Wie der TX beherrscht der ftDuino den Standard (100 kHz) und den Fast Mode (400 kHz). Damit können alle zum TX kompatiblen I2C-Sensoren und -Aktoren auch an den ftDuino angeschlossen werden.
- Sogar der Ultraschall-Abstandssensor wird vom ftDuino unterstützt, allerdings ausschließlich am Zählereingang C1.

Das verträgt sich leider nicht mit einer Ansteuerung aus ROBO Pro.

Fünf Abweichungen vom TX bleiben: Der ftDuino

- beherrscht (bisher) keine Synchronisierung der Encodermotoren³
- unterstützt keine serielle Bluetooth-Verbindung als USB-Alternative,
- besitzt keine Taster unter dem Display,
- kann nur PWM-Werte von 0 bis 64 verarbeiten (TX: 0-512) und
- unterstützt keine Extensionen.⁴

Das sind aber für die wenigsten Anwendungen essentielle Einschränkungen. Daher sollte es möglich sein, einen ftDuino aus ROBO Pro im Online-Mode via USB fast wie einen TX zu steuern.

Das Fish.X1-Protokoll

Ein Sketch, der auf dem ftDuino das Verhalten eines TX „simuliert“, muss auf Kommandos von ROBO Pro genauso reagieren wie ein TX. Dazu muss er dieselbe „Sprache“ sprechen wie ROBO Pro, sprich: das von ROBO Pro verwendete Protokoll beherrschen.

Die Kommunikation zwischen ROBO Pro und dem TX erfolgt über ein von den Entwicklern „Fish.X1“ getauftes Protokoll. Eine Übersicht des generellen Aufbaus der Pakete findet sich in der Dokumentation von MSC [14]; es fehlen allerdings (vollständige) Dokumentationen

- der Kommandos, die ROBO Pro im Online Mode an den TX sendet,
- des Aufbaus der vom Protokoll mit den Kommandos verschickten Datenpakete (die „Payload“ des Protokolls) und

- des Ablaufs des Protokolls zwischen ROBO Pro und dem TX.

Dieses Wissen ist erforderlich, um einen „TX-Simulator“ auf dem ftDuino korrekt auf die Datenpakete von ROBO Pro reagieren zu lassen, wie beispielsweise das Setzen einer Spannung an einem Motorausgang oder das Auslesen eines Eingangswerts.

Informationsrecherche

Einen Teil der benötigten Informationen kann man dem Header-File des C-Programmierspakets, das MSC mit dem TX ausgeliefert hat, und dem Anhang der zugehörigen Dokumentation entnehmen [15] – wie z. B. den grundsätzlichen Aufbau der Fish.X1-Protokollpakete (Abb. 2). Da dort aber in erster Linie die Programmierschnittstelle der ftMscLib dokumentiert wird, fehlen viele Angaben zum Fish.X1-Protokoll.

In Ermangelung einer Dokumentation des Herstellers, die im Übrigen auch fischertechnik nicht vorliegt, hilft da nur ein „Reverse Engineering“ des Fish.X1-Protokolls. Thomas Kaiser analysierte 2009 mit hohem technischen Aufwand die RS-485-Verbindung zwischen dem TX und den angeschlossenen Extensionen, die ebenfalls das Fish.X1-Protokoll verwendet [16]; wenig später ergänzte Christoph Niessen Thomas' wertvolle Erkenntnisse um eigene Analysen [17].

Ende 2012 schließlich widmete sich ft-ninja dem Protokoll, um einen Parser für den Arduino zu entwickeln. Seine Fortschritte kann man im Forum nachlesen [6]. Raphael Jakob steuerte 2018 in der ft:pedia einige weitere Einsichten über das Protokoll bei und dokumentierte die bis dahin bekannte Struktur der Datenpakete [18].

³ Die Erweiterung der ftDuino-Funktionsbibliothek um einen Synchron-Befehl sollte prinzipiell möglich sein.

⁴ Über I2C-Verbindungen könnte man weitere ftDuinios als Slaves nutzen und so Extensionen simulieren.

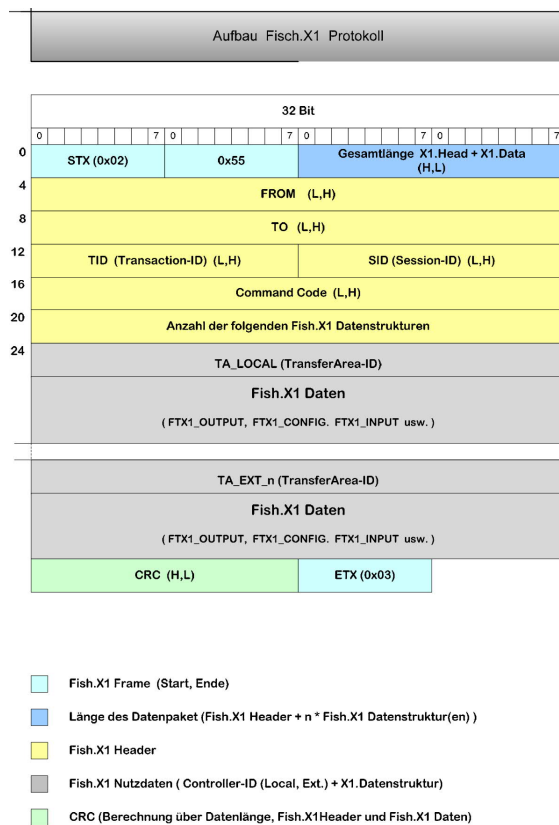


Abb. 2: Aufbau eines Fish.X1-Protokoll (Quelle: MSC [15])

Die Ergebnisse der verschiedenen Analysen stimmen jedoch nicht in allen Details überein. Sie beschränken sich zudem auf die einfache Motorsteuerung und das Einlesen von Inputs. Es fehlen Hinweise zur Ansteuerung des I2C-Protokolls, die fischertechnik im März 2012 mit dem Update der TX-Firmware auf Version 1.30 eingeführt hatte [19].

Reverse Engineering

Zur Überprüfung der unterschiedlichen Angaben und um eine möglichst komplette Übersicht aller von ROBO Pro genutzten Fish.X1 Command Codes inklusive der zugehörigen Datenpakete zu erhalten, erstellte ich zu allen ROBO-Pro-Befehlen, die im Online-Mode eine Übertragung an den TX vornehmen, kleine Testprogramme.

Dann schnitt ich die zwischen ROBO Pro und dem TX während des Programmablaufs ausgetauschten Datenpakete mit einem Protokoll-Sniffer⁵ mit (Abb. 3). Durch kleine Variationen der Testprogramme konnte ich über die jeweiligen Änderungen in den Datenpaketen den Aufbau und die Bedeutung der Einträge rekonstruieren bzw. bereits dokumentierte Strukturen bestätigen.

Abb. 3: Mitschnitt des Fish.X1-Protokolls (blau markiert: CMD 002-Paket, I/O Request an Master und Extension 8)

Die Ergebnisse der Analyse habe ich in einer separaten Dokumentation zusammengefasst, die ich nach Fertigstellung zum Download bereitstellen werde. Dabei waren die bereits existierenden Dokumentationen natürlich eine sehr große Hilfe – so konnte ich mich auf die fehlenden und uneinheitlich dargestellten Teile des Protokolls konzentrieren.

Im Folgenden stelle ich den Protokollablauf mit einem TX ohne Extensionen vor.⁶

Der Protokollablauf

Nach der Auswahl des Controllers baut ROBO Pro zunächst via USB (UART) eine

⁵ Wireshark 3.6.8 mit USBPcap 1.5.4.

⁶ Der Ablauf mit angeschlossenen Extensionen ist deutlich komplexer.

serielle Verbindung zum TX mit einer Übertragungsgeschwindigkeit von 38.400 Baud (Bit/s), 8 Datenbits, 1 Stopp-Bit und ohne Parity-Bit auf. Der Timeout wird auf 2000 ms gesetzt.

Wird ein Programm im Online-Mode oder der Interface-Test gestartet, fordert ROBO

Pro via Broadcast eine Fish.X1-Verbindung an (Abb. 4).⁷ Die Session ID der Verbindung legt der TX fest, sobald er den Verbindungsaufbauwunsch (*Connect Request*, CMD 001-Paket) von ROBO Pro erhalten hat, und gibt sie im Antwortpaket (*Connect Reply*, CMD 101-Paket) zurück.

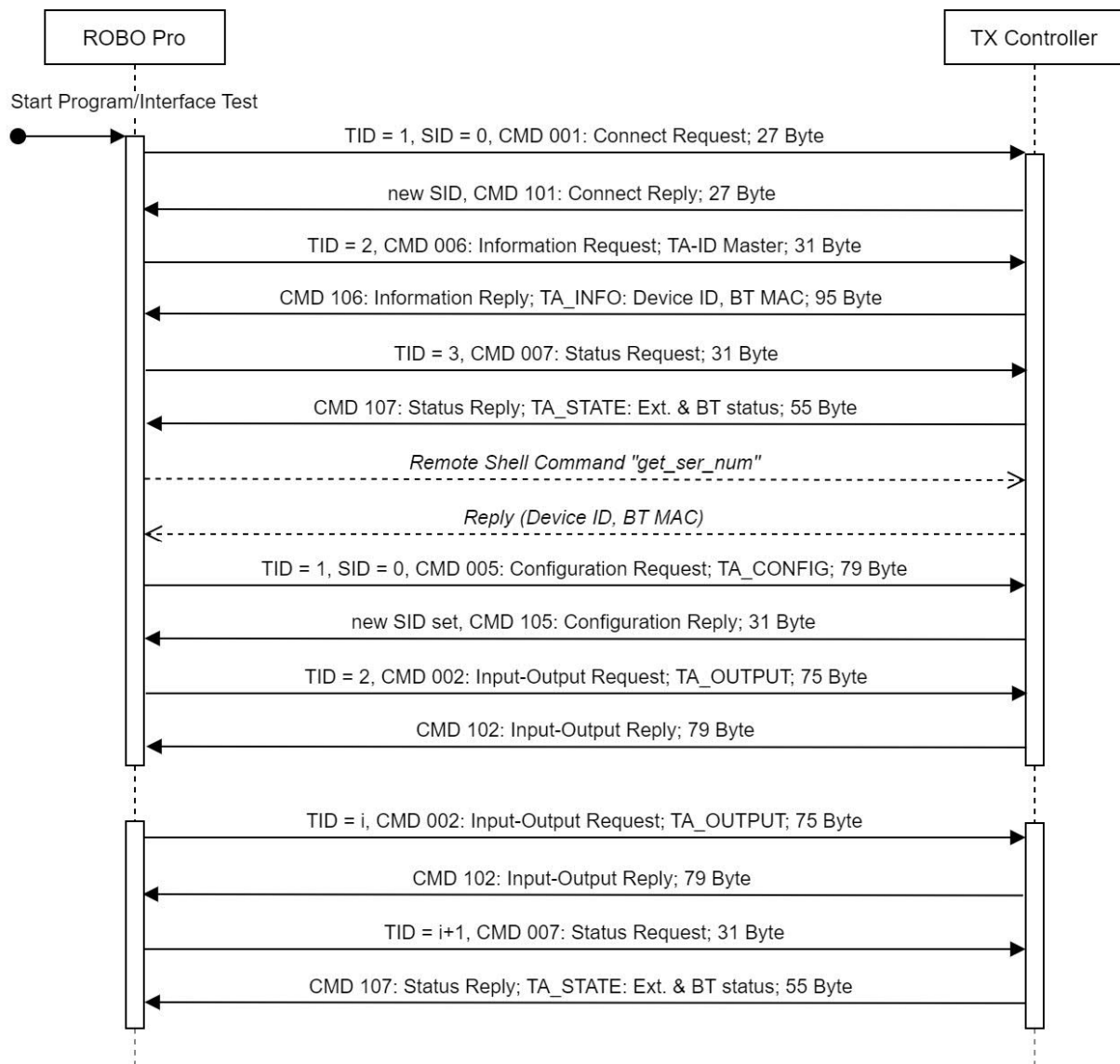


Abb. 4: Fish.X1-Protokollablauf zwischen ROBO Pro und einem TX ohne Extensions (TID: Paketnummer, SID: Verbindungsnummer, CMD 00x: Befehlspaket)

⁷ Auch für alle folgenden Pakete verwendet ROBO Pro im Fish.X1-Frame die Broadcast-Adresse 0x0001 als TO-Adresse. Vermutlich

wird damit sichergestellt, dass das Paket auch dann den Master erreicht, wenn der PC mit einer Extension verbunden ist.

Bei der ersten Verbindung nach dem Einschalten des TX ist die SID = 1. Es folgt ein *Information Request* (CMD 006-Paket), der den Namen des TX (Device ID) und dessen Bluetooth-MAC-Adresse anfordert.

Daran schließt sich ein *Status Request* an (CMD 007-Paket), der nach den angeschlossenen Extensionen und BT-Verbindungen fragt.

Nun kennt ROBO Pro die Konfiguration und beginnt eine neue Fish.X1-Verbindung, in der gleich im ersten Paket die Konfiguration der Input- und Output-Anschlüsse vorgenommen wird (*Configuration Request*, CMD 005-Paket). Der TX vergibt im Antwortpaket (CMD 105) eine neue SID.

Danach geht ROBO Pro in einen festen Rhythmus über und schickt alle 0,01 Sekunden einen *Input-Output* (CMD 002-Paket) und jede Sekunde einen *Status Request* (CMD 007-Paket), um Änderungen bei den angeschlossenen Extensionen festzustellen.

Dieser Ablauf wird nur von I2C-Lese- (CMD 019-Paket) und -Schreibbefehlen (CMD 020-Paket) unterbrochen, wenn diese im ROBO Pro-Programm vorkommen.

Das Fish.X1-Protokoll endet, wenn sowohl das ROBO Pro-Programm als auch der Interface-Test beendet oder gestoppt wurden.

Der „TX-Simulator“

Ausgangspunkt: Der fx1parser

Der von ft-ninja für den Arduino entwickelte Fish.X1-Parser ist sehr ausgefeilt und stabil. Er erkennt die 5 wichtigsten der 7 von ROBO Pro genutzten Fish.X1-Datenpakete zuverlässig und antwortet auch auf das von ROBO Pro eingeschobene Remote-Shell-Kommando korrekt (siehe Abb. 4).

Für den Arduino konnte ft-ninja einige Vereinfachungen vornehmen: Die Ein- und Ausgänge musste er nicht konfigurieren, da technisch feststeht, welche Ports analog und welche nur digital genutzt werden können.

Zählereingänge gibt es beim Arduino nicht und daher auch keine erweiterte Motorsteuerung (Abstandsbehl).

Anpassungen an den ftDuino

Ein „[TX-Simulator](#)“-Sketch für den ftDuino erforderte, den Parser um die folgenden Funktionen zu erweitern:

- Konfiguration der Input- und Output-Ports sowie der schnellen Zählereingänge (CMD 005-Pakete)
- Auslesen und Zurücksetzen der Zählereingänge (CMD 002-Pakete)
- Implementierung der erweiterten Motorsteuerung (Abstandsbehl)
- Erkennung von I2C-Steuerpaketen durch den Parser und
- Implementierung der I2C-Steuerung (CMD 019- und CMD 020-Pakete)

Ein paar kleine Optimierungen konnte ich ebenfalls am Parser vornehmen:

- Da keine Extensionen unterstützt werden, liegt die maximale Länge der Fish.X1-Datenpakete bei 95 Byte. Begrenzt man die drei Puffer (rx/tx/tmp) entsprechend, spart man 483 Byte des knappen RAM-Speichers.
- Die Bearbeitung des Empfangspuffers erfolgt Byte für Byte, daher verwendet der „TX-Simulator“ einen globalen Pointer, anstatt die Pointer jeweils als Funktionsparameter zu übergeben.
- Der Parser prüft jedes empfangene Zeichen. Am häufigsten wartet er dabei auf weitere Daten im Paket, am zweithäufigsten auf das Paket-Anfang-Signal (STX). Daher sollten diese beiden Zustände vom Parser zuerst überprüft werden.
- Auch die Bearbeitung eines CMD-Pakets lässt sich ein wenig beschleunigen, wenn die am häufigsten auftretenden Befehle (CMD 002 und CMD 007) zuerst abgefragt werden.

Die Anpassung an die Befehle der ftDuino-Bibliothek [20] ist nicht schwierig, da die Programmierschnittstelle bis auf die I2C-Befehle alle benötigten Controller-Kommandos als fertige Funktionen bereitstellt.

Implementierung

Bei der Implementierung des „TX-Simulators“ waren ein paar Besonderheiten zu beachten und Entwurfsentscheidungen zu treffen.

Controller Status (CMD 007)

Mit einem *Status Request* (CMD 007) fragt ROBO Pro zu Beginn des Protokolls die Zahl der an einen TX angeschlossenen Extensionen und aktiven Bluetooth-Verbindungen ab. Diese Abfrage wird einmal pro Sekunde wiederholt.

Da der „TX-Simulator“ weder Extensionen noch Bluetooth-Verbindungen am ftDuino unterstützt, wird dieser Request immer mit einem „leeren“ Paket (Status: offline) beantwortet.

I/O-Konfiguration (CMD 005)

Mit einem *Configuration Request* (CMD 005) gibt ROBO Pro die Nutzung der Ein- und Ausgänge des TX vor. Diese Einstellungen können im Verlauf eines ROBO-Pro-Programms nicht geändert werden, daher schickt ROBO Pro sie gleich zu Beginn des Protokollablaufs an den TX (Paket 1 der zweiten Verbindung, s.o.).

Die Einstellungen umfassen

- die jeweils paarweise Aktivierung der 8 Output Ports als Motorausgang,
- den Typ der Input Ports (Widerstand/ Spannung, analog/digital, Abstand) und
- die Zählmethode der Counter (steigende/fallende Flanke).

Bei den Motorausgängen muss sich der „TX-Simulator“ lediglich merken, welche aktiv sind und nicht als separate Output Ports genutzt werden.

Die Input Ports werden am ftDuino mit der Methode

```
ftduino.input_set_mode(port, type)
```

konfiguriert. Dabei unterscheidet der ftDuino zwischen SWITCH (digital, Widerstand), RESISTANCE (analog, Widerstand) und VOLTAGE (analog, Spannung).

Bei Anschluss eines fischertechnik-Spursensors erwartet ROBO Pro abhängig von der anliegenden Spannung einen binären Wert (0/1). Da die Eingänge des ftDuino nicht auf digital/Spannung konfiguriert werden können, stellt der „TX-Simulator“ einen solchen Eingang auf VOLTAGE (analog) und merkt sich, dass der Eingang für einen Spursensor konfiguriert wurde.

Sehr einfach funktioniert die Einstellung der Zählmethode für die Counter:

```
ftduino.counter_set_mode(cnt, type)
```

Dabei legt der Typ fest, ob bei jeder (C_EDGE_ANY), nur bei fallender Flanke (C_EDGE_FALLING) oder nur bei steigender (C_EDGE_RISING) gezählt wird. Als Voreinstellung ist der Zähler beim ftDuino deaktiviert (C_EDGE_NONE). ROBO Pro setzt den Wert standardmäßig auf Zählung bei fallender Flanke.

Universaleingänge (CMD 002)

Alle 10 ms fragt ROBO Pro mit einem *Input Output Request* (CMD 002) die an den – mit dem vorausgegangenen *Configuration Request* eingestellten – Unversaleingängen anliegenden Werte ab. Dafür bietet die ftDuino-Bibliothek eine einfache Methode:

```
ftduino.input_get(port)
```

Der zurückgelieferte Wert ist bei

- einer Spannungsmessung ein Wert von 0 bis 10.000 (= 10 V)
- einer Widerstandsmessung ein Wert von 0 bis 65535 (Ohm); ab ca. 10 kOhm werden die Werte immer ungenauer [20]
- einem (digitalen) Schalter 0 oder 1

Ist ein Eingang für einen Spursensor konfiguriert (digital, Spannung), vergleicht der „TX-Simulator“ den gemessenen Analogwert mit einem Schwellenwert: Ist dessen Wert größer, wird 0 zurückgeliefert, andernfalls 1. Als Schwellenwert verwendet der „TX-Simulator“ 4000 (ca. 4V); diese Konstante kann bei Bedarf im Sketch angepasst werden:

```
#define LINE_SENSOR_THRESHOLD 4000
```

Zählereingänge (CMD 002)

Mit jedem *Input Output Request* (CMD 002) werden auch die Zählereingänge abgefragt. Der TX liefert sowohl den jeweiligen Zählerstand als auch das am Zählereingang anliegende Potential (high, low) zurück. Beim ftDuino werden diese Werte für jeden Zählereingang mit den Methoden

```
ftduino.counter_get(cnt)
ftduino.counter_get_state(cnt)
```

ausgelesen.

Dass ein Zählereingang zurückgesetzt werden soll, teilt ROBO Pro dem TX über einen erhöhten Zählerwert mit. Um die Änderung des Zählerwerts erkennen zu können, muss sich der „TX-Simulator“ daher den jeweils letzten Zählerstand merken. Die Rücksetzung eines Zählers erfolgt dann mit der Methode

```
ftduino.counter_clear(cnt)
```

Die erfolgte Rücksetzung des Zählers wird ROBO Pro zusammen mit den Werten der Universaleingänge und Zählerwerte im Antwortpaket *Input Output Reply* (CMD 102) durch ein einmalig gesetztes Flag (`cnt_resetted[cnt]`) gemeldet.

Motorsteuerung (CMD 002)

Mit einem *Input Output Request* (CMD 002) übergibt ROBO Pro die PWM-Werte für die Ausgänge O1 bis O8. Dafür „übersetzt“ ROBO Pro Motorgeschwindigkeit und Drehrichtung eines Motorausgangs in PWM-Werte für die beiden zugehörigen Output Ports. Diese Werte muss der „TX-

Simulator“, sofern der Motorausgang als „aktiv“ konfiguriert wurde, für den ftDuino in Drehrichtung und PWM-Wert zurückrechnen. Dabei sollte die Drehrichtung des Motors der beim TX entsprechen.

Der von ROBO Pro übermittelte PWM-Wert hat immer eine Auflösung von 9 bit (1-512), auch wenn der Motor in ROBO Pro auf die Werte 1-8 (3 bit) eingestellt wurde. Die Umrechnung der Werte nimmt ROBO Pro vor. Für den ftDuino muss der „TX-Simulator“ die empfangenen PWM-Werte auf die Auflösung der Motortreiber des ftDuino (6 bit, 1-64) umrechnen. Das erledigt der hilfreiche „map“-Befehl der Arduino-IDE.

Die Motorausgänge des ftDuino werden danach mit der Methode

```
ftduino.motor_set(mot, mode, pwm)
```

und die Ausgangs-Ports mit der Methode

```
ftduino.output_set(port, mode, pwm)
```

eingestellt.

Mit den mit dem TX eingeführten erweiterten Motorbefehlen in ROBO Pro werden die Encodermotoren gesteuert. Damit kann ein Motor für eine vorgegebene Anzahl Encoder-Impulse aktiviert werden.

Um den Encoder auszuwerten, werden schnelle Zählereingänge benötigt. Sowohl beim TX als auch beim ftDuino bewältigen sie bis zu 1 kHz – das entspricht bei den Encodermotoren der ersten Generation (75 Impulse je Umdrehung) etwas mehr als 780 U/min und bei den aktuellen Encodern (63,333... Impulse je Umdrehung) knapp 950 U/min. Beide Werte liegen deutlich über der Leerlaufdrehzahl der Encoder-Motoren (280-325 U/min).

Einen erweiterten Motorbefehl signalisiert ROBO Pro über eine Erhöhung des Zählerwerts `cnt_ext_motor_cmd_id[mot]`. Die ftDuino-Bibliothek kennt als Entsprechung zum Abstandsbefehl in ROBO Pro die Methode

```
ftduino.motor_counter(mot, mode, pwm,
    distance)
```

Als Abstand wird wie in ROBO Pro die Zahl der Encoder-Impulse übergeben. Damit wird zugleich der Motor gestartet.⁸

Anders als der TX, der zunächst den Zähler auf 0 zurücksetzt, setzt der ftDuino für die Abstandsmessung den Zählerstand auf die negative Distanz und zählt bis 0.

Die Methode

```
ftduino.motor_counter_active(mot)
```

verrät, ob der Motor die Position bereits erreicht hat (`false`). Hat er, dann stoppt der Motor automatisch. Damit der Motor möglichst wenig nachläuft, muss der Motor-Modus `brake` eingestellt sein – das ist beim ftDuino die Default-Einstellung:

```
ftduino.motor_counter_set_brake(mot,
    on)
```

Der „TX-Simulator“ muss das Erreichen der Position mit dem nächsten *Input-Output Reply* (CMD 102) an ROBO Pro zurückmelden. Dazu erhöht er den Zähler `motor_pos_reached[mot]` um eins.

Nach einem erweiterten Motorbefehl ignoriert der TX einfache Motorbefehle so lange, bis der Zustand mit einem Abstandsbefehl über die Distanz „0“ aufgehoben wird. Daher muss sich der „TX-Simulator“ diesen Zustand merken. Auch für den abschließenden Abstandsbefehl muss er den Zähler `motor_pos_reached[mot]` um eins erhöhen und das Flag `cnt_resetted[cnt]` setzen.

Zu beachten ist, dass das *Input-Output-Request*-Paket von ROBO Pro gleich mehrere Motor-Befehle enthalten kann – so

z. B. einen Abstandsbefehl mit „0“ und eine Geschwindigkeitsvorgabe für einen normalen Motorbefehl.

I2C-Kommandos (CMD 019/020)

Die Fish.X1-Datenpakete *I2C Write Request* (CMD 020) und *I2C Read Request* (CMD 019), die mit dem Update der ROBO Pro-Version 3.1.3 und des TX auf die Firmware-Version 1.30 vom 19.03.2012 eingeführt wurden, sind nirgendwo dokumentiert.

Sie fallen auch etwas aus dem Rahmen: Obwohl I2C-Befehle in ROBO Pro direkt einer angeschlossenen Extension zugewiesen werden können, enthalten die Fish.X1-Datenpakete keine Adresse, sondern nur die für den I2C-Befehl relevanten Daten. Und während im Fish.X1-Protokoll ansonsten sehr großzügig mit Datenfeldern umgegangen wird⁹, werden die I2C-Konfigurationsdaten in ein Byte gequetscht (Tab. 1).¹⁰

Beim *I2C Write Request* (CMD 020) werden neben der 7-bit-I2C-Adresse und dem Konfigurations-Byte (Tab. 1) bis zu 4 Byte Daten übergeben, die der „TX-Simulator“ abhängig von der Konfiguration an ein am ftDuino angeschlossenes I2C-Device übertragen muss.

Bei einem *I2C Read Request* (CMD 019) werden – abhängig von der Konfiguration – bis zu zwei Byte („Unteradresse“) an das I2C Device geschickt, bevor ein bis zwei Byte gelesen werden. Bei beiden Kommandos kann der I2C-Bus „offen“ gehalten und damit reserviert, d. h. für andere Devices gesperrt werden.

⁸ Leider werten TX und ftDuino den Encoder nur bei steigender (ftDuino) bzw. bei fallender Flanke (TX) aus. Würden beide Trigger ausgewertet, ließe sich damit die Auflösung der Encoder verdoppeln.

⁹ So wurden z. B. für die FROM- und TO-Adressen jeweils vier Byte spendiert – obwohl von den

damit möglichen 4.294.967.296 Adressen lediglich 9 benötigt werden...

¹⁰ Das könnte damit zusammenhängen, dass die USB-Verbindung mit 38 kBit/s nur ein Zehntel des Datendurchsatzes des vom TX unterstützten I2C Fast Mode (400 kBit/s) erreicht.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
[0]0	100 kHz	ein Versuch		Close	-		Keine Unteradr.	
[0]1	400 kHz	10 Versuche		Open	8 bit Daten		8 bit Unteradr.	
10	-	bis erfolgreich		-	16 bit Daten		16 bit Unteradr.	
11	-	-		-	-		-	

Tab. 1: Aufschlüsselung des I2C-Konfigurations-Byte

Für die I2C-Kommandos auf dem ftDuino muss die Bibliothek „wire.h“ eingebunden werden. Das Schreiben von Daten an ein bestimmtes I2C-Device erfolgt nach dem Öffnen einer Verbindung bytewise mit einem Write-Befehl:

```
Wire.beginTransmission(i2c_addr);
Wire.write(data);
Wire.endTransmission();
```

Das Lesen von Daten kann ebenfalls sehr einfach bytewise erfolgen:

```
Wire.requestFrom(i2c_addr);
while (Wire.available() < 0);
data = Wire.read();
```

Die empfangenen Daten werden in einem *I2C Read Reply* (CMD 119) an ROBO Pro zurückgeliefert.

Als Default-Einstellung verwendet der ftDuino den I2C Standard Mode (100 kHz). Die Umstellung auf den Fast Mode erfolgt (bei geschlossener Verbindung) mit

```
Wire.setClock(400000);
```

Damit können auf dem ftDuino auch I2C-Sensoren genutzt werden (sofern ein ROBO Pro Treiber existiert), ohne dass ein Sensor-Treiber in den knappen Flash-Speicher des ftDuino geladen werden muss.

Tests

Eine der besonderen Eigenschaften von ROBO Pro ist der Interface-Test, der alle an den Ein- und Ausgängen anliegenden Werte des TX (bzw. in unserem Fall des einen TX simulierenden ftDuino) anzeigt – und der sogar gestartet werden kann, während gleichzeitig ein Programm im Online-Mode läuft. Damit lassen sich bei den erweiterten

Motorbefehlen beispielsweise die Werte der schnellen Zählereingänge oder bei den Universaleingängen Änderungen des Spursensors live mitverfolgen – eine große Hilfe bei der Fehlersuche oder Programmoptimierung. So kann man auch das Funktionieren des „TX-Simulators“ mit dem Interface-Test leicht überprüfen. Für Tests der I2C-Kommandos muss man auf die in den meisten ROBO Pro-Treibern enthaltenen Beispielprogramme zurückgreifen, weil der Interface-Test dazu keine Daten anzeigt. Eine umfangreiche Sammlung mit insgesamt 27 Treibern für verbreitete I2C-Sensoren findet sich in meinem Github-Account [zum Download](#) [21].

Ergebnis

Der [TX-Simulator für den ftDuino](#) täuscht ROBO Pro einen TX Controller vor. Mit nur wenigen Einschränkungen (keine Extensionen, keine Synchronisation von Encoder-Motoren, keine Bluetooth-Kommunikation, kein Ultraschall-Abstandssensor) kann der Befehlsumfang von ROBO Pro im Online-Mode nahezu vollständig mit dem ftDuino genutzt werden – inklusive I2C-Sensoren, dem Oszillografen und der erweiterten Motorsteuerung für Encoder-Motoren. Damit erweitern sich die Einsatzmöglichkeiten des ftDuino vor allem in Schulen:

Als Einsteigerprogrammiersprache senkt ROBO Pro die Einstiegshürde, können dieselben ROBO Pro-Programme (mit wenigen Ausnahmen) sowohl auf dem TX(T) als auch auf dem ftDuino genutzt werden, und ermöglicht die Oszilloskopfunktion von ROBO Pro die Nutzung des ftDuino als Messgerät in (Physik-) Experimenten.

Ausblick

Über den I2C-Bus könnte man weitere ftDuinos als „Slaves“ ansteuern – die sich so via „[TX-Simulator](#)“ wie Extensionen aus ROBO Pro steuern ließen. Auch die Synchronisation von Encodermotoren ließe sich bei Bedarf auf dem ftDuino realisieren.

Falls ROBO Pro in größerem Umfang auf ftDuinos zum Einsatz kommen sollte, ist eine entsprechende Erweiterung des „TX-Simulators“ nicht ausgeschlossen... Rückmeldungen sind daher sehr willkommen.

Quellen

- [1] Torsten Stuehn: *ROBOPro, ftrobopy und ftScratch auf dem TXT 4.0*. [ft:pedia 2/2022](#), S. 123–129.
- [2] Dirk Fox: *Endliche Automaten in Robo Pro*. [ft:pedia 3/2014](#), S. 42–50.
- [3] Till Harbaum: *Von Lucky Logic zu RoboPro Coding*. [ft:pedia 1/2021](#), S. 103–109.
- [4] Till Harbaum: *Brickly auf dem TXT: Grafische Programmierung à la Google-Blockly*. [ft:pedia 1/2017](#), S. 92–98.
- [5] Peter Habermehl: *startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi*. [ft:pedia 1/2018](#), S. 102–107.
- [6] ft-ninja: [ROBO Pro + Arduino](#). Thread im Forum der ft:community.
- [7] mr-kubikus: [fx1-arduino-parser](#). Github, zuletzt aktualisiert (Test mit ROBO Pro 4.6.6) am 19.02.2021.
- [8] Thomas Magin: *Die Kunst der H-Brücke: Let's Rock*. [ft:pedia 2/2022](#), S. 104–118.
- [9] Dirk Fox, Thomas Püttmann: *fischertechnik-Roboter mit Arduino*. dpunkt-Verlag, 2000.
- [10] Holger Howey: *TX Light – Arduino Uno, der von RoboPro gesteuert wird*. Bilderpool der ft:community, 18.02.2017.
- [11] Holger Howey: *TX-Light: Arduino (Uno/Mega) und ftduino aus ROBO Pro ansteuern*. [ft:pedia 3/2022](#), in dieser Ausgabe.
- [12] Till Harbaum: *ftduino – Open-Source trifft Konstruktions-Baukasten*. [ft:pedia 1/2018](#), S. 85–91.
- [13] Till Harbaum: [ftduino – Arduino für fischertechnik](#). github.io.
- [14] Peter Duchemin: *Programmierung des ROBO TX Controllers. Teil 1: PC-Programmierung*. V1.5, 24.04.2012.
- [15] Peter Duchemin: *Programmierung des ROBO TX Controllers. Teil 2: Windows Library „ftMscLib“*. V1.5.11, 4.04.2012.
- [16] Thomas Kaiser: [Die Robo-TX RS-485 Schnittstelle](#). 17.12.2009.
- [17] Christoph Nießen: [Die RS-485-Schnittstelle des ROBO TX Controllers](#). 2010 (?).
- [18] Raphael Jacob: *Programmierung des TX-Controllers mit Python*. [ft:pedia 2/2018](#), S. 60–67.
- [19] Dirk Fox: *I²C mit TX und Robo Pro – Teil 1: Grundlagen*. [ft:pedia 3/2012](#), S. 32–37.
- [20] Till Harbaum: [ftduino – ein fischertechnik-kompatibler Arduino](#). Bedienungsanleitung. 12.01.2021. Github.io
- [21] Dirk Fox: [I2C-Treiber für ROBO Pro](#). 27 ROBO Pro-Treiber für verbreitete I2C-Sensoren. github.com